Universidad
Carlos III de Madrid

TRABAJO FIN DE GRADO

Título: Sistema de Gestión de Red para Vehículos Aéreos no Tripulados
Autor: Patricia Díaz Muñoz
Titulación: Grado Ingeniería Telemática
Tutor: Iván Vidal Fernández
Co-Director: Francisco Valera Pintor
Fecha: Julio 2016

Agradecimientos

A mis padres.

Resumen

El proyecto que se desarrolla en este Trabajo de Fin de Grado consiste en el diseño e implementación de un sistema de gestión de red en vehículos aéreos no tripulados, también conocidos con su término en inglés UAV (*Unmanned Aerial Vehicles*). Estos dispositivos son cada vez más utilizados en la actualidad para diversas tareas, como por ejemplo, tareas de rescate en zonas de difícil acceso, vigilancia, etc.

En primer lugar, para comenzar el proyecto, se hará un estudio de la gestión de red, donde se explique qué es, qué conceptos incluye y cómo éstos se entrelazan entre sí para poder llegar a lograr el cometido de vigilar, gestionar y monitorizar una red.

A partir de este punto, se realizará un estudio de las herramientas de gestión de red más populares actualmente, considerando sus características, ventajas y desventajas entre ellas, creando una comparativa entre ellas para finalmente decidir dos que se implementarán en un despliegue preliminar de evaluación.

Este primer despliegue ayudará para comprender y familiarizarse con las herramientas elegidas de forma que al trabajar con ellas, se puedan valorar las ventajas y desventajas entre ellas y de este modo, elegir una para el diseño y desarrollo final.

Con los resultados obtenidos del despliegue preliminar, el diseño de la solución se basará en las necesidades del despliegue real, de forma que, tras probar las dos herramientas en el despliegue preliminar, se elija cuál de las dos se adecúa mejor a este diseño y posteriormente, aplicarlo al desarrollo.

Finalmente, la implementación de la herramienta nos permitirá obtener datos sobre la red creada para trabajar y esta implementación será validada a través de un conjunto de pruebas para comprobar su correcto funcionamiento en este despliegue.

Palabras clave: monitorización de red, gestión de red, pruebas, diseño, desarrollo, despliegue

Índice general

Índice de figuras

Índice de tablas

Acrónimos y abreviaturas

Acrónimo o abreviatura	*Significado*
AGC	*Automatic Gain Control*
BDS	*Berkeley Software Distribution*
BW	*Bandwidth*
CDEF	*Cython Declared Functions*
CGI	*Common Gateway Interface*
COIT	*Colegio Oficial de Ingenieros de Telecomunicación*
CPU	*Central Processing Unit*
DAU	*Data Access Unit*
DHCP	*Dynamic Host Configuration Protocol*
E/S	*Entrada/Salida*
FTP	*File Transfer Protocol*
GCS	*Ground Control Station*
GNU	*GNU's Not Unix*
GPL	*General Public License*
HDMI	*High-Definition Multimedia Interface*
HW	*Hardware*
IP	*Internet Protocol*
IPv6	*Internet Protocol version 6*
ISO	*International Organization for Standardization*
LAN	*Local Area Network*
LOPD	*Ley Orgánica de Protección de Datos*
MIB	*Management Information Base*
NMS	*Network Monitoring System*
NOC	*Network Operation Center*
OID	*Object ID*
OS	*Operating System*

PC	*Personal Computer*
Perl	*Practical Extraction and Report Language*
POSIX	*Portable Operating System Interface UniX*
RFC	*Request For Comments*
RRA	*Round Robin Archive*
RRD	*Round Robin Database*
RRD (2)	*Reduced Resolution Dataset-File*
RRDTool	*Round Robin Database Tool*
RTT	*Round Trip Time*
SMNP	*Simple Network Management Protocol*
SMNPv3	*Simple Network Management Protocol version 3*
SQL	*Structured Query Language*
SSH	*Secure SHell*
TCP	*Transmission Control Protocol*
UAV	*Unmanned Aerial Vehicle*
URL	*Uniform Resource Locator*
XML	*eXtensible Markup Language*

Capítulo 1

Introducción

1.1 Motivación

Hoy en día, los seres humanos estamos muy acostumbrados al concepto de "red". Ya sea la red local doméstica de una casa, o la red móvil a la que se conecta el móvil cuando se sale fuera de ella, la cuestión es que las redes siempre están presentes en nuestras vidas.

Teniendo esto en cuenta, es normal que nosotros pensemos en desarrollar nuevas ideas para poner en práctica en las redes. Pongamos por ejemplo una situación peligrosa como un accidente de coche, un alpinista perdido en la montaña, un terremoto ... Hoy en día, el tiempo de llegada de ayuda es reducido pero podría ser mejor. Con el avance de las nuevas tecnologías además del desarrollo de las redes, podría ser posible mejorarlo gracias a herramientas como la monitorización de red proactiva. Aquí es donde ya podemos tener en cuenta el UAV (*Unmanned Aerial Vehicle*) y el concepto de red.

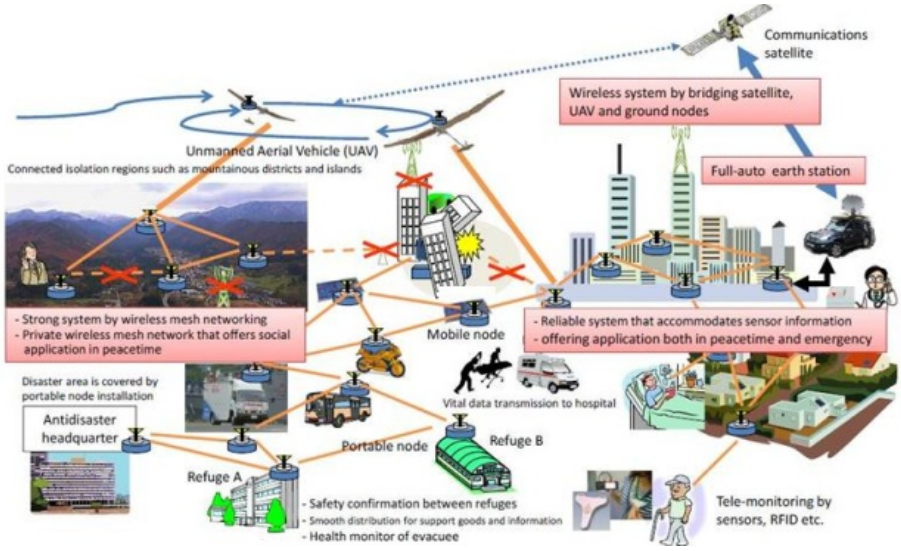


Figura 1. Red de malla con UAV, satélites y nodos terrestres [UAV Network, 2016]

Con los UAV, además de otros componentes de red, se pueden desarrollar grandes redes de malla para cubrir y vigilar ciertas áreas, por ejemplo, áreas que sean propensas a terremotos, para evitar situaciones peligrosas proactivamente o, cuando el desastre ya haya ocurrido, resolverlo y ayudar a los posibles afectados tan pronto como sea posible [UAV Network, 2016].

Por otro lado, también hay que tomar en cuenta que es normal que las redes tengan problemas ocasionales, debido a su constante uso. Las redes están compuestas de gran cantidad de componentes hardware y software, de los cuáles algunos de ellos son complejos, y como consecuencia, pueden dejar de funcionar o funcionar mal, o incluso resultar recursos sobreutilizados.

Como resultado de esto, es necesario que alguien o “algo” gestione la red, para asegurarse de que funciona correctamente y si hay algún problema, tratar de resolverlo proactivamente - es decir, antes de que ocurra - o reactivamente, según se necesite. Esto es posible gracias a los servicios y herramientas de las que dispone la gestión y monitorización de la red, un concepto realmente importante para este proyecto.

1.2 Objetivos

Como se ha podido ver, la gestión y monitorización de la red es un concepto fundamental que puede ser bastante utilizado hoy en día y en el futuro sobre todo, ya que se usará para trabajar proactivamente con las redes existentes.

El principal objetivo de este Trabajo de Fin de Grado es llevar a cabo el despliegue, configuración y validación de un sistema de gestión de red para entornos de vehículos aéreos no tripulados, con el propósito de recoger datos y estadísticas de los dispositivos con capacidad de comunicación que existen en dichos entornos, para después procesarlos, almacenarlos y mostrarlos gráficamente.

Este objetivo principal se divide a su vez en los siguientes sub-objetivos:

- Analizar el estado del arte de las principales herramientas de gestión de red que existen en el mercado, haciendo una comparativa entre sus ventajas y desventajas, además de describir las características que cada una posee y que pueden hacerlas potencialmente las idóneas para el entorno considerado en este TFG.
- Teniendo en cuenta el resultado de dicho análisis, realizar una evaluación práctica preliminar de aquellas herramientas que se valoren como más apropiadas para un entorno de UAVs, probándolas en un despliegue sencillo sobre el que se configurará una recogida de datos y se mostraran gráficamente los resultados obtenidos.
- Como resultado de los dos sub-objetivos anteriores, se seleccionará una única herramienta, que constituirá la base del sistema de gestión de red cuyo despliegue se plantea en el presente TFG. El sistema de gestión de red se configurará y particularizará para su apropiado funcionamiento en un entorno de UAVs, y el despliegue realizado se validará mediante diversas pruebas.

- Finalmente, se valorarán diversas opciones para la ejecución de los componentes del sistema de gestión de red y comprobar la versatilidad de la herramienta elegida, en particular las siguientes: (1) ejecución sobre un equipo dedicado; (2) ejecución mediante un sistema de máquinas virtuales; (3) y ejecución sobre una plataforma portable de tamaño reducido como una Raspberry Pi.

Como objetivo secundario a la realización de este TFG, aunque no menos importante, se pretende también adquirir conocimientos y experiencia en diversas tecnologías relacionadas con las redes.

1.3 Estructura de la memoria

La estructura que se ha seguido para escribir este Trabajo de Fin de Grado es análoga a la que está explicada en la Matriz de Evaluación del Trabajo de Fin de Grado. Se puede decir que está dividida en dos secciones principales.

Por un lado:

- En primer lugar, se ha visto una introducción con las motivaciones y objetivos que se esperan conseguir en esta investigación, siendo la principal la posibilidad de nuevos avances en la tecnología gracias a la gestión y monitorización de red, y en concreto, en una aplicación particular, los UAV.
- Posteriormente, algunos objetivos se establecerán para desplegar un sistema de monitorización de red con herramientas conocidas en un despliegue UAV.
- Además, habrá una subsección donde se explicará el estado del arte. En esta sección, se van a explicar algunos conceptos importantes relacionados con la gestión de red además de hacer una pequeña investigación sobre las herramientas de monitorización actuales, útiles para hacer una tabla comparativa después donde se verán las ventajas y desventajas de cada una. Así, finalmente se elegirán dos de ellas para implementar en un desarrollo de pruebas. En esta sección, también se verán las restricciones que tenemos que tener en cuenta y el marco regulador vigente bajo el que vamos a trabajar.

Por otro lado, la otra sección principal que hemos mencionado con anterioridad es la que podríamos considerar la “práctica”. En esta sección se pueden encontrar los siguientes objetivos:

- Se estudiará el diseño y desarrollo de la solución - con su relativa implementación - además de las pruebas llevadas acabo para garantizar la eficacia de la herramienta elegida.

Para finalizar, habrá otra sección con la planificación del Trabajo de Fin de Grado y su presupuesto correspondiente, creada a partir de las necesidades de recursos que necesitemos. También habrá una sección de conclusiones donde se hablará de las posibles futuras líneas de trabajo en las que se explican las dificultades y conocimientos contenidos con esta

investigación, además de cómo se usarán en un futuro para desarrollar nuevas ideas. Para terminar, se podrán consultar las secciones con los anexos y los recursos bibliográficos utilizados.

NOTA: Para la lectura del apartado Introducción en inglés, se puede consultar el Anexo VII – apartado Introduction

Capítulo 2

Estado del arte

2.1 Introducción

En primer lugar, la gestión y monitorización de la red es un componente muy importante a la hora de hacer un despliegue de red, ya que es la herramienta principal con la que se controla el correcto funcionamiento de dicha red y la resolución de posibles problemas.

A continuación, se hará un planteamiento del tema a tratar en este proyecto, que es establecer los precedentes a la monitorización de red, explicando cómo este concepto va ligado al proyecto que se tiene en mente y cómo aplicando distintas herramientas se puede llegar a crear un sistema de gestión de red para un vehículo aéreo no tripulado. Posteriormente, se explicarán algunas de las herramientas de gestión y monitorización de red más importantes en la actualidad, estudiando sus ventajas y desventajas - así como sus características - a través de una tabla comparativa. Finalmente, se elegirán un par de ellas que serán probadas en el apartado de Diseño e Implementación a modo de ver cuál será la finalmente elegida para nuestro proyecto a través de un despliegue preliminar.

2.2 Planteamiento del problema

A la hora de desarrollar un problema relacionado con la gestión de red, hay que hacerse las siguientes preguntas: ¿qué es la gestión de red?, ¿para qué sirve?, ¿qué conceptos dependen de la gestión de red?

Es por ello que es importante estudiar estos conceptos en primer lugar, y después ver cómo estos se entrelazan para que lleguemos a entender lo que significa la gestión de una red y cómo este conocimiento ayudará a realizar el despliegue deseado en el caso que ocupa.

2.2.1 Gestión de red

La gestión de red, como se ha mencionado con anterioridad, es un concepto primordial en el que se basa este proyecto, ya que la monitorización de red es una de las características básicas de la gestión de red. Básicamente, la gestión de red es la responsable de la monitorización, gestión y control de la red, sus componentes, herramientas y servicios, de forma proactiva y reactiva [Kurose & Ross, 2013]. Algunos de los conceptos que se van a explicar a continuación son necesarios para entender la gestión de red.

Infraestructura de la red

Uno de los componentes más importantes de la gestión de red es su infraestructura [Millán Tejedor, 1999; Kurose & Ross, 2013]. A continuación, se muestran los componentes de una infraestructura de gestión de red y sus cometidos:

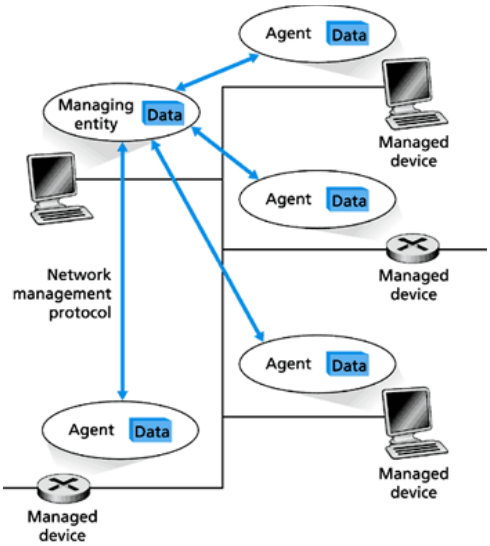


Figura 2. Network Management Infraestructure [Network Management, 2016]

Entidad gestora (Managing entity)

Es la aplicación, con frecuencia dependiente de intervención humana, que se ejecuta en una estación central de gestión de red situada en el *Centro de Operaciones de Red* (en inglés NOC, *Network Operation Center*). Es el punto focal de la actividad de administración de red; controla la recopilación, procesamiento, análisis y/o visualización de la información de gestión de la red. También es donde se inician las acciones para el control de comportamiento de la red y donde el administrador interactúa con los dispositivos que forman la red.

Dispositivo gestionado (Managed device)

Es un equipo de red (*host*, *router*, *brigde*, *switch*, …) que se desea gestionar. Dentro de él pueden existir varios objetos gestionados, es decir, los propios elementos hardware del dispositivo y/o los conjuntos de parámetros de configuración para los distintos elementos que

componen dicho hardware y software. Estos objetos gestionados tienen asociados distintos elementos de información que se recopilan dentro de un tipo de base de datos llamada *Base de Información de Gestión* (en inglés MIB, *Management Information Base*). En cada dispositivo, también existe un agente de gestión de red, que es un proceso que se ejecuta en el dispositivo gestionado y que se comunica con la entidad gestora y los dispositivos gestionados, permitiendo a aquélla consultar el estado de los dispositivos gestionados bajo el control de la entidad gestora.

Protocolo de gestión de red

Se ejecuta entre la entidad gestora y los dispositivos gestionados, permitiendo a aquélla consultar el estado de los dispositivos gestionados y llevar a cabo acciones y eventos. No se encarga él mismo de administrar la red; simplemente proporciona una serie de capacidades que un administrador puede utilizar para gestionar la red. En definitiva, es como el "puente" que une al gestor con los dispositivos que se desean controlar, permitiendo que "hablen un idioma común".

Simple Network Management Protocol, SNMP

Específicamente, el protocolo más común usado para la gestión de red es SNMP, también conocido como *Simple Network Management Protocol* [RFC 3410, 2002; SNMP, 2015]. Éste es un protocolo de nivel de aplicación que sirve primordialmente para el intercambio de información de administración entre dispositivos de red. La versión más actual del protocolo es SNMPv3 [Millán Tejedor, 2003], la cual posee mejoras y novedades respecto a versiones anteriores, tanto en características como en seguridad, aspecto muy importante.

Entre sus componentes, se hallan:

- *Sistema administrador de red (Network Management System, NMS),* también conocido como la **entidad gestora**, que se encarga de ejecutar aplicaciones para el control y supervisión de los dispositivos administrados.
- *Dispositivo gestionado*, que es el dispositivo perteneciente a la red sobre el que se va a llevar las tareas de administración.
- *Agente*, que es un módulo de software de administración de red residente en el dispositivo. Se encarga de conocer la información local del dispositivo administrado y traducir luego al formato SNMP dicha información para poder formar el árbol de jerarquía que será después accesible en el MIB.

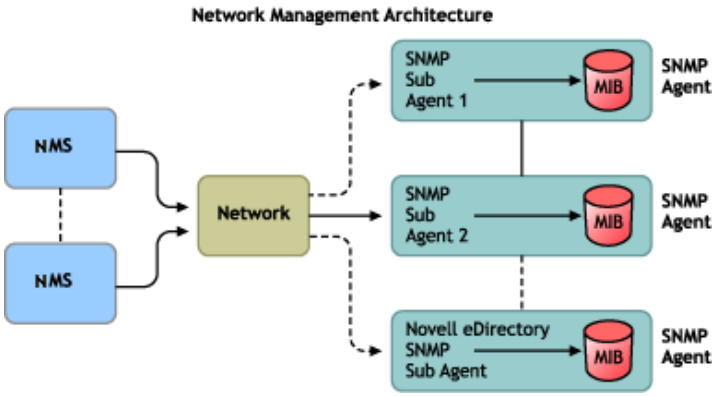


Figura 3. Arquitectura de gestión de red con SNMP [SNMP, 2016]

Management Information Base, MIB

Otro concepto a tener en cuenta cuando hablamos del protocolo SNMP es el de MIB (*Management Information Base*) [Kurose & Ross, 2013].

Como se ha mencionado anteriormente, el MIB es una base de información que ordena jerárquicamente por categorías las características propias del dispositivo administrado, a las cuales se va accediendo ordenadamente gracias a su estructura, y que dan lugar a los llamados objetos MIB. Cada objeto MIB puede ser identificado con un identificador único, llamado *Object ID* (*OID*).

Finalmente, la versión actual del protocolo, la versión 3, incluye mejoras con respecto a anteriores versiones tanto como en la administración remota como en seguridad, haciendo el protocolo mucho más eficaz.

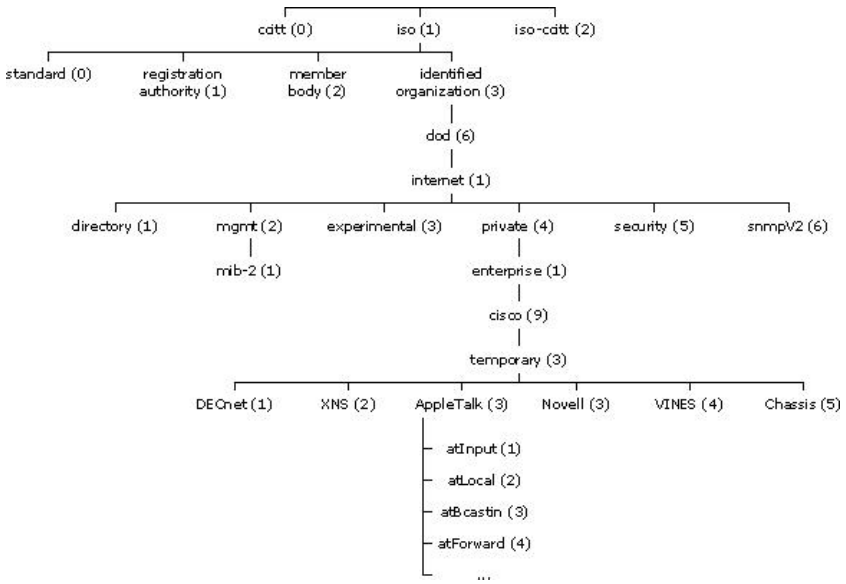


Figura 4. Representación parcial de un árbol MIB-II [Wikipedia, 2015]

Eventos de red gestionados

Se han visto hasta aquí los componentes de una red gestionada y cómo se comunican. Sin embargo, es necesario mencionar también qué eventos pueden ser gestionados por la gestión de red. Como se puede ver en las figuras 2 y 3, existe un vínculo establecido entre el gestor y los dispositivos gestionados gracias a los protocolos de gestión de red como SNMP, haciendo posible que se gestionen eventos de red [Kurose & Ross, 2013]. Algunos de estos eventos son, por ejemplo: detección de fallos de una tarjeta de interfaz en un host o router, monitorización de los hosts, detección de intrusiones …

Modelo de gestión de red ISO

Además, para hacer todo más ordenado, la gestión de red se engloba dentro de las reglas ISO (*International Organization for Standardization*) y se propone un modelo a seguir para su correcto funcionamiento, también conocido por *FCAPS*, por las siglas que conforman sus características, que en español se traducen a gestión de fallos, de configuración, de cuentas, de rendimiento y de seguridad [Kurose & Ross, 2013].

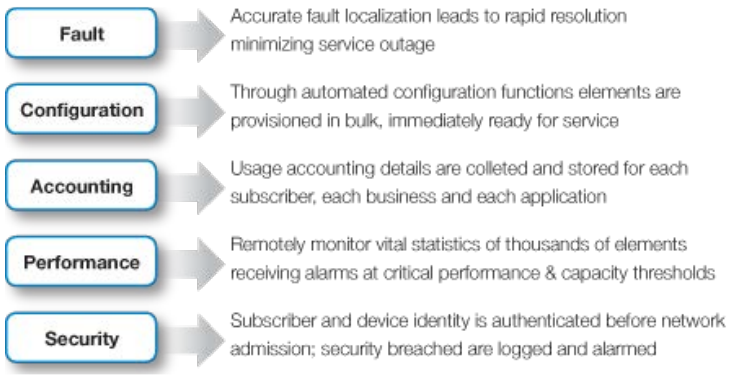


Figura 5. Modelo FCAPS [Selesta Networks, 2015]

Como consecuencia, se necesitan estudios sobre la gestión de red en UAV, no sólo porque serán útiles en posibles situaciones de peligro, pero también porque ayudarán a resolver los problemas antes. El desarrollo de un sistema de gestión de red para estos UAV es indispensable para monitorizar y gestionar la red en la que operan, ya que es la principal fuente de estudio de problemas e información de la que dispondremos.

Es por ello que, en la actualidad, existen multitud de herramientas para llevar a cabo la monitorización de la red de forma sencilla e intuitiva. Gracias al estudio de algunas de estas herramientas, se verá cuál se acopla mejor a las necesidades del entorno que hay que desarrollar.

2.3 Análisis de herramientas de gestión de red

Al llegar a este punto, ya se conocen los términos principales que se agrupan bajo la gestión de red. Ahora es necesario estudiar las principales ventajas y desventajas de las herramientas de gestión de red actuales para elegir la que mejor convenga para nuestro proyecto.

Se estudiarán las tres herramientas a priori más populares del mercado para este despliegue – habiendo hecho un descarte con anterioridad de la multitud existente. De estas tres herramientas, se verán sus características y efectividad, las cuáles se compararán al final, eligiendo dos de las tres posibles para implementar en primera instancia en un despliegue de pruebas.

2.3.1 CACTI – The complete rrdtool-based graphing solution

Introducción

Cacti [Barry et al., 2013; The Cacti Group Inc., 2004-2012] es una de las herramientas elegidas para probar su funcionalidad en el sistema de gestión de red de vehículos aéreos no tripulados. Es una solución de gestión de red basada en las funcionalidades gráficas y de almacenamiento de datos de RRDTool [Oetiker, 2014], mejorándolas y dándolas un enfoque más intuitivo para el usuario. Es una herramienta *Open Source* y de software libre, dos características fundamentales.

Además de eso, Cacti está diseñada para ser una herramienta *frontend* que permite almacenar e introducir datos en sus bases de datos para crear gráficos más versátiles y completos. También es capaz de recoger los datos y manejarlos de forma sencilla. Esto es posible gracias a la creación de scripts o comandos externos a la herramienta que permiten al usuario recoger todo el tipo de datos que quiera. Por ejemplo, si se desean conocer las estadísticas de un sensor determinado del vehículo aéreo no tripulado, se puede crear un script que recoja esas estadísticas y que Cacti lo procese para crear gráficas y que el usuario pueda ver cómo funciona ese sensor a partir de dicha gráfica.

Cacti también consta de un sistema de administración de usuarios para permitir a ciertos usuarios la gestión y manipulación de la configuración de los gráficos para propósitos personales. Es importante también el uso de plantillas, que permite que Cacti sea una herramienta de gestión de red escalable y por ello, cuando se añade un nuevo *host* a la red, sea posible recoger información de dicho dispositivo gracias a las plantillas creadas.

Características

A continuación se va hacer un breve resumen de las características fundamentales de la herramienta de gestión de red Cacti.

Gráficos

- Se puede definir un número ilimitado de elementos gráficos para cada gráfico utilizando opcionalmente CDEF’s (comandos de creación de datos) o fuentes de datos de Cacti.

- Agrupación automática de elementos gráficos para hacer una rápida re-secuenciación de los mismos.
- Los datos gráficos se pueden manipular usando funciones matemáticas CDEF que vienen en RRDTool. Estas funciones CDEF pueden venir definidas en Cacti y pueden usarse globalmente en cada gráfico.
- Soporta todo tipo de elementos gráficos de RRDTool.

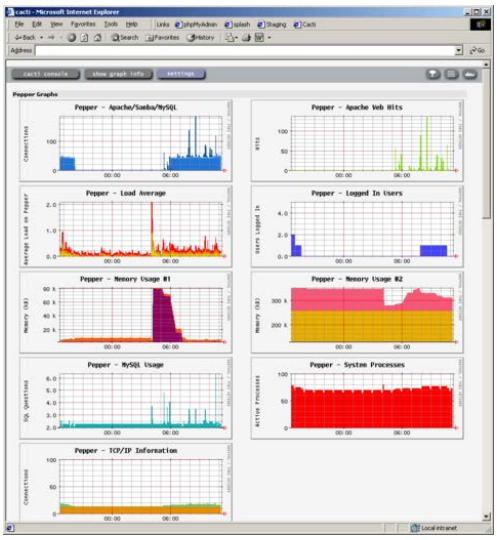


Figura 6. Árbol de gráficos Cacti [The Cacti Group Inc., 2004-2012]

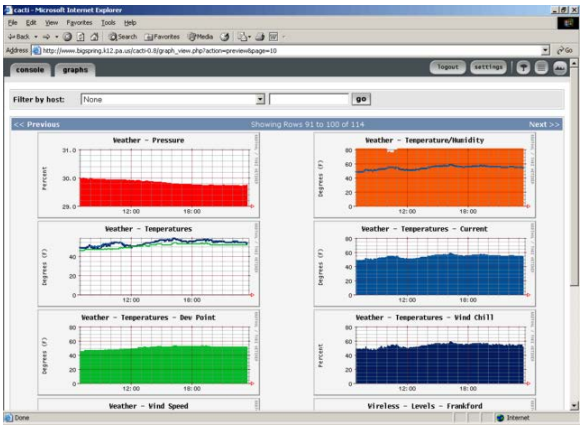


Figura 7. Gráficos en modo *preview* [The Cacti Group Inc., 2004-2012]

Fuentes de datos

Las fuentes de datos se refieren a los parámetros o ítems de los cuáles se recogerán estadísticas.

- Las fuentes de datos se pueden crear usando los comandos RRDTool “create” y “update”. Cada fuente de datos se puede usar para recoger datos de forma local o remota, para después visualizarlos en una gráfica.
- Soporta archivos RRD (*Reduced Resolution Dataset-File*) con más de una fuente de datos y puede usar un archivo RRD almacenado en el sistema.
- La configuración de los archivos RRA (*Round Robin Archive*) puede personalizarse permitiendo al usuario la habilidad de representar datos en distintos intervalos de tiempo mientras se almacenan grandes cantidades de datos.

Recogida de datos

En este apartado se pueden ver los principales mecanismos para recoger estadísticas de los que dispone Cacti.

- Contiene un mecanismo de “data input” que permite a los usuarios definir scripts personalizados que se pueden usar para obtener datos.
- Soporta SNMP, por lo tanto Cacti tiene la habilidad de recoger datos SNMP.

Para realizar la recogida de datos, ya sea a través de scripts o sean datos SNMP, se utiliza un *poller* PHP, que además sirve para actualizar los archivos RRD.

Plantillas

Se pueden crear plantillas con Cacti para evitar tener que crear desde cero si se desea aplicar los mismos mecanismos de recogida de datos y/o gráficos en distintos dispositivos administrados. Así mismo, las plantillas de Cacti pueden ser de:

- Gráficos
- Fuentes de datos
- *Hosts*

Exposición de gráficos

En la exposición de gráficos se pueden ver las distintas formas de mostrar los gráficos en Cacti:

- La vista de árbol permite a los usuarios crear "jerarquías gráficas" y poner gráficos en el árbol. Esta es una manera sencilla de administrar/organizar un gran número de gráficos.
- La vista "list" lista el título de cada gráfico en una gran lista que enlaza al usuario con el gráfico actual.
- La vista "preview" expone todos los gráficos en el formato de una gran lista de gráficos.

Administración de usuario

Además de todas las características anteriores, Cacti permite llevar a cabo una administración de la herramienta por parte del usuario de forma que puede:

- Permitir a los administradores crear usuarios y asignar los diferentes niveles de permisos para la interfaz Cacti. Estos permisos pueden ser especificados en cada gráfico por cada usuario.
- Cada usuario puede guardar sus propias configuraciones de gráficos para preferencias diversas.

Principios de operación

Cacti [Barry et al., 2013] funciona de la siguiente manera, dividiendo su trabajo en tres grandes áreas:

Figura 8. Principios de operación de Cacti

Recogida de datos

El primer paso de Cacti es la recogida de los datos que se van a representar en las gráficas. Esta función la realiza a través de un elemento llamado *Poller*, el cuál es ejecutado desde el planificador del sistema operativo.

En la actualidad, al utilizar muchos dispositivos basados en la gestión de red, se pueden recoger los datos a través del uso del protocolo SNMP, soportado también por Cacti.

Almacenamiento de datos

El almacenamiento de los datos obtenidos [Barry et al., 2013] es el siguiente paso a seguir con Cacti. Los datos pueden ser almacenados de diversas formas (MySQL, flat files, …) pero Cacti usa RRDTool. Éste está basado en el almacenamiento de datos temporales usando el concepto de *Round Robin* – el tamaño del "almacén" es constante, los datos nuevos sobrescriben a los antiguos, haciendo que dicho "almacén" nunca crezca en tamaño.

Además usando RRDTool, también se hace uso de las llamadas funciones de consolidación, que combina los datos obtenidos a lo largo del tiempo con datos ya almacenados, comprimiendo el espacio y haciendo un uso más eficaz de él.

Presentación de datos

Por último, la presentación de los datos obtenidos por Cacti se hace a través de los gráficos, muy útiles para que el usuario pueda ver de forma intuitiva y sencilla como se comporta el sistema gestionado. La creación de gráficas puede variar en función de los deseos del usuario y de los datos que se quieran representar, así que Cacti cuenta con diversas funcionalidades para hacer esto posible.

Spine

Spine [The Cacti Group Inc., 2004-2012] es una funcionalidad de Cacti que actúa como un *poller* de dicha herramienta para funcionar tan rápido como se pueda. Está escrito en lenguaje C y usa las hebras de POSIX, haciendo posible el trabajo asíncrono y mejorando la velocidad de recogida de los datos. También está directamente enlazado con la librería **net-snmp** para reducir la sobrecarga de *polling*. El uso de Spine es recomendable según el tamaño de la red, siendo muy útil en aquellas que sean de tamaño grande, ya que el *poller* funciona más rápidamente.

2.3.2 Zabbix – The Enterprise-class Monitoring Solution for Everyone

Introducción

Zabbix [Zabbix SIA, 2015; Zabbix, 2015] es una de las herramientas de gestión de red más completas e innovadoras que existen actualmente. Está orientada principalmente a grandes negocios que constan de un conjunto de redes grande y necesariamente escalable, aunque también es útil para pequeñas y medianas empresas. Ofrece multitud de servicios todos compactados en una única solución, haciendo sencillo el uso de esta herramienta, además de ofrecer mejoras dentro de periodos cortos de tiempo para que los usuarios encuentren a

Zabbix como una herramienta segura y fiable para gestionar sus redes y sus infraestructuras tecnológicas.

Las funcionalidades de Zabbix son las típicas de una herramienta de gestión de red: recogida, almacenamiento y administración de datos, monitorización web, alertas y gráficos. Lo que la hace realmente especial es que es una solución única, condensando todos sus servicios en una única solución. Además es totalmente *Open Source*, puede monitorizar cualquier cosa – como entornos distribuidos - y al estar centrada en grandes empresas , goza de gran escalabilidad, donde en un solo nodo Zabbix se pueden concentrar 25000 hosts y ofrecer chequeos de 100 métricas de hosts cada minuto.

Por otro lado, Zabbix pretende mejorar los problemas que aparecen a la hora de gestionar una red, como por ejemplo, periodos de inactividad, administración de entornos no transparentes o la planificación de la red.

En definitiva, Zabbix es una herramienta no sólo de gestión de red, sino también de grandes infraestructuras tecnológicas con gran cantidad de servicios bastante bien implementados y con una interfaz amigable para el usuario, ofreciendo los datos que se desean de una forma organizada y con una visualización excelente.

Características

A continuación se van a explicar brevemente las principales características de la segunda herramienta de gestión de red, Zabbix.

Monitorización de todo tipo de dispositivos

Zabbix es capaz de monitorizar todo tipo de dispositivos, desde servidores hasta aplicaciones, obteniendo datos y estadísticas concretos. Entre todo lo que monitoriza Zabbix se puede encontrar:

- Monitorización de rendimiento
- Monitorización de parámetros sin necesidad de agente
- Dispositivos de red
- Monitorización de parámetros personalizados a través de scripts
- Monitorización distribuida
- Bases de datos
- Aplicaciones Java
- Monitorización Hardware

Además, Zabbix es capaz de crear mapas de red para monitorizar la infraestructura disponible.

Orientación a entornos empresariales

La principal característica de Zabbix a la hora de hablar de su orientación a entornos empresariales es su escalabilidad. Esta herramienta puede ser utilizada tanto para pequeños despliegues de red – empresas pequeñas y medianas – como para grandes compañías que poseen conjuntos de redes grandes de miles de dispositivos que han de ser monitorizados.

Este nivel de escalabilidad es posible gracias al uso de algoritmos muy eficientes y muy bien diseñados, los cuáles aprovechan la modularidad actual del hardware y el software que divide los componentes en varios servidores, logrando un resultado más eficiente.

Debido a esta característica, Zabbix posee servicios principalmente orientados a ella, tales como la monitorización distribuida a través de elementos que actúan como *proxies*, mantenimiento de la herramienta nulo – pues lo único que hay que hacer es ir actualizándola y esto no supone cambios drásticos – además de componentes de seguridad, fácil implementación y preparado para IPv6 – garantizando el uso en el futuro de esta herramienta.

Monitorización proactiva

La monitorización proactiva consiste en conocer de antemano los sucesos que van a acontecer en la red para que en un despliegue se puedan anticipar problemas u optimizar recursos, consiguiendo de este modo reducir los costes operativos, evitar problemas como periodos de inactividad y como no, mejorar la calidad de servicio.

Zabbix hace posible esto gracias a diversos mecanismos y servicios que tiene para llevar a cabo la monitorización proactiva, tales como:

- Alertas
- Manejadores de eventos
- Comentarios para poder dar pistas sobre fallos o problemas
- Información útil sobre el dispositivo
- Soluciones técnicas por parte de especialistas
- Capacidad de resolver incidencias de forma rápida

Planificación de capacidad

Gracias a la posibilidad de la recogida de datos, estos se pueden analizar para prever en cuánto aumentará la capacidad de una red y con ello, evitar problemas que afecten a la escalabilidad de la red. Con Zabbix esto es posible y es una gran característica que muy pocas herramientas de gestión de red poseen.

Soluciones de negocio

Al ser una herramienta orientada a grandes empresas, Zabbix ofrece multitud de soluciones de negocio útiles para aquellas organizaciones que quieran hacer un uso de dicha herramienta más extenso. Entre estas soluciones se encuentran:

- Soporte comercial
- Desarrollo personalizado
- Solución *Turn-key* (se despliega e implementa la herramienta en muy poco tiempo)
- Cursos de formación oficiales
- Consultoría

Gráficas

Los modos de visualización de Zabbix son muy extensos, así como la creación de gráficas. Se pueden crear desde gráficas sencillas con un par de parámetros de entrada hasta gráficas más complejas – como gráficas personalizadas con muchos más parámetros o gráficas *ad-hoc*, creadas para comparar multitud de dispositivos de forma eficiente.

Todas ellas tienen muchas posibilidades para crear gráficos orientativos y fáciles de leer para el usuario. Algunos ejemplos son las siguientes figuras (de la 9 a la 12):

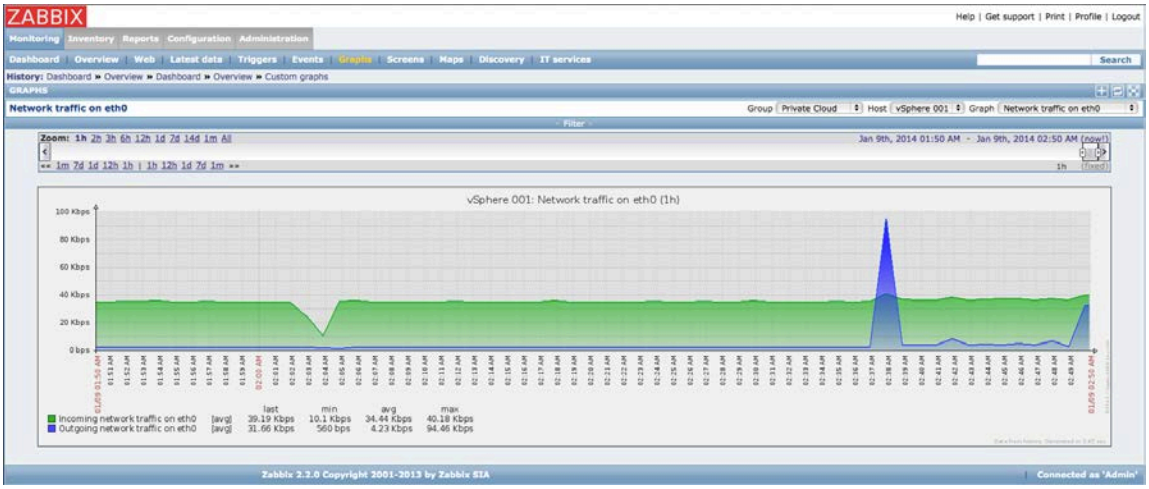


Figura 9. Gráfico de uso de red [Zabbix SIA, 2015; Zabbix, 2015]

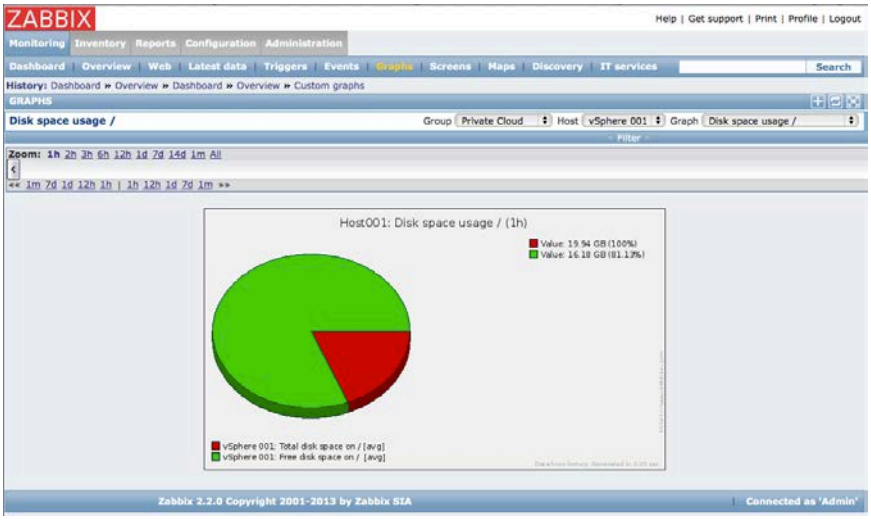


Figura 10. Gráfico de uso de disco [Zabbix SIA, 2015; Zabbix, 2015]

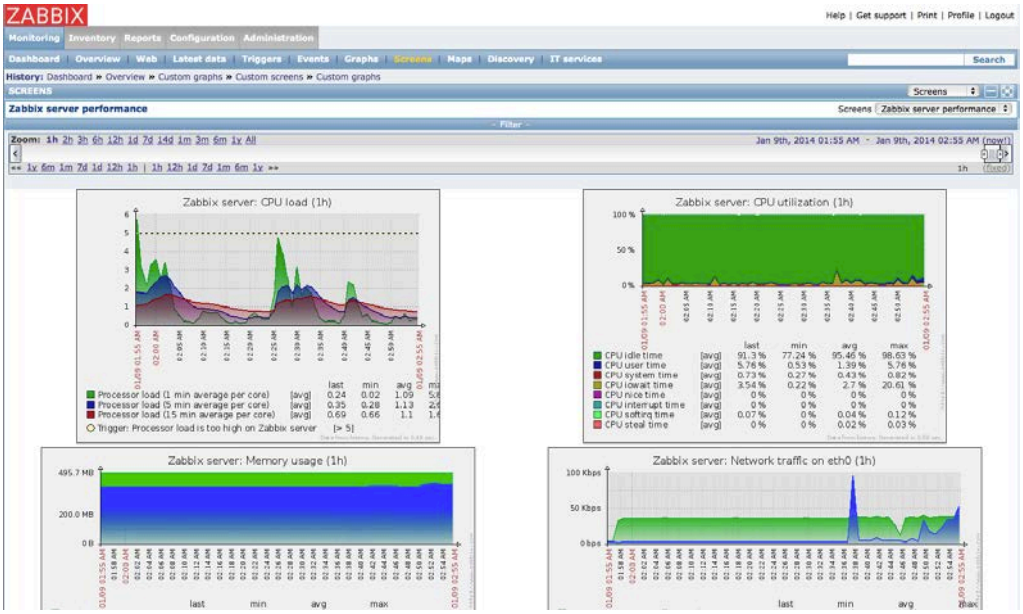


Figura 11. *Screens*: carga de CPU, uso de CPU, uso de memoria … [Zabbix SIA, 2015; Zabbix, 2015]

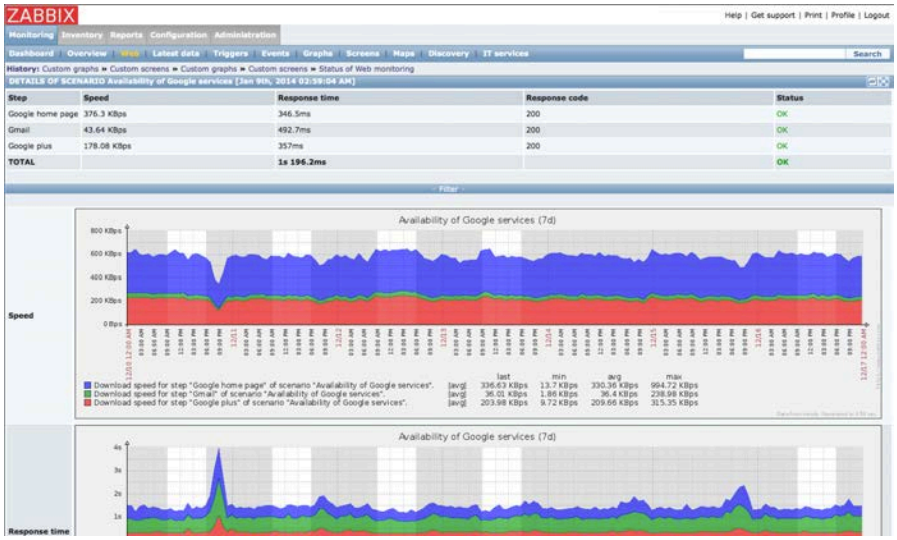


Figura 12. Gráficos de servicios web y datos recogidos [Zabbix SIA, 2015; Zabbix, 2015]

Plantillas y scripts

Como se ha mencionado con anterioridad, Zabbix es una herramienta de gestión de red que tiene entre sus principales características su escalabilidad y su orientación a negocios. Es por ello que el uso de plantillas y scripts es una funcionalidad muy importante dentro de ella.

Las plantillas sirven para crear patrones equivalentes de estudio y/o monitorización de los dispositivos que son imprescindibles a la hora de incorporar nuevos elementos a la red, pues aplicándoles dichas plantillas, será posible llevar a cabo la monitorización de dichos nuevos dispositivos.

Por otro lado, también se ha mencionado antes la capacidad de Zabbix de ofrecer a sus usuarios un contenido y monitorización personalizados. Esto es posible gracias a la creación de scripts que recogen estadísticas de forma personalizada que luego pueden ser visualizadas en los gráficos creados con Zabbix.

2.3.3 Monitorix

Introducción

La última herramienta de gestión de red que se va a examinar se llama Monitorix [Sanfeliu, 2001-2015]. Es una herramienta de monitorización sencilla diseñada para el control y monitorización de una gran cantidad de dispositivos. Es libre y *Open Source*, dos características imprescindibles para nuestro estudio. Está diseñado para monitorizar servidores Linux/UNIX principalmente, pero gracias a su simplicidad puede ser incorporado en otros dispositivos.

La funcionalidad de Monitorix se basa en dos programas: un colector llamado **monitorix**, consistente en un demonio escrito en Perl que arranca automáticamente como un servicio del sistema y un script CGI (*Common Gateway Interface*), llamado **monitorix.cgi**. Cada vez que **monitorix** arranca, lee un fichero de configuración del path especificado en el comando y una vez que lo ha leído, crea el *index.html* para mostrar en forma de página web los gráficos obtenidos. También se crea un fichero llamado *monitorix.conf.path* que incluye el path al fichero de configuración. Este fichero será el que lea **monitorix.cgi** para determinar la localización del fichero de configuración.

Monitorix es una herramienta desarrollada sobre RRDTool, así que los gráficos que produzca serán archivos **rrd** y serán tratados y manejados al igual que los archivos que se manejan con RRDTool.

Actualmente Monitorix sigue desarrollando nuevas características, gráficas y mejoras para hacer de este servicio uno de los más importantes a nivel de la administración de sistemas.

Características

A continuación se van a explicar brevemente las principales características de la última herramienta de gestión de red, Monitorix.

Visualización de un gran rango de tipo de gráficos

Como herramienta de gestión de red, Monitorix es capaz de producir gráficos para la visualización de los datos recogidos y su representación. Además, es capaz de representar un gran rango de variables y estadísticas de los datos recogidos en gráficos completos, como son, entre otras:

- Carga y uso del sistema
- Uso global del kernel
- Uso del kernel por procesador
- Temperaturas de sensores, tarjetas gráficas, discos duros…
- Uso del sistema de archivos y actividad de E/S

- Estadísticas de procesos
- Estadísticas de correo
- Uso y tráfico de red
- Tráfico del puerto de red

Algunos ejemplos son las siguientes gráficas (de la figura 13 a la 16):

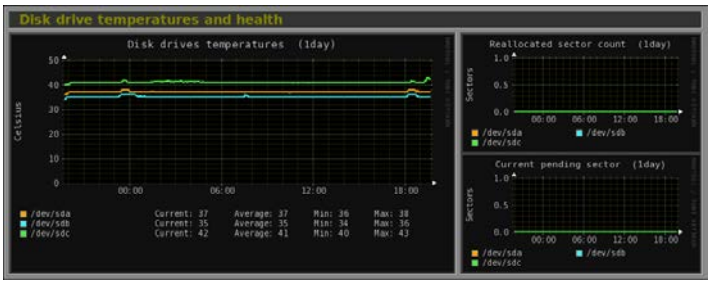


Figura 13. Temperaturas de disco duro [Sanfeliu, 2001-2015]

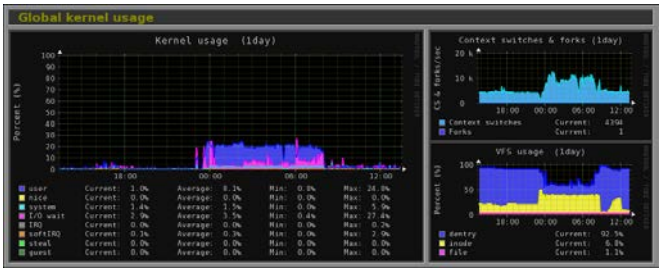


Figura 14. Uso global del *kernel* [Sanfeliu, 2001-2015]

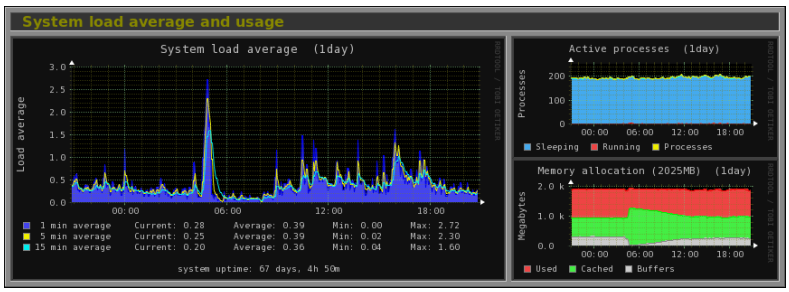


Figura 15. Uso y carga media del sistema [Sanfeliu, 2001-2015]

Figura 16. Estadísticas de procesos [Sanfeliu, 2001-2015]

Soporte para monitorización remota y del tráfico de Internet a través de dispositivos LAN

La monitorización remota consiste en llevar a cabo la monitorización de dispositivos ajenos, es decir, pertenecientes a una red que contrata tus servicios de monitorización. Ésta es una práctica muy usada recientemente en todo tipo de entornos empresariales, haciéndoles la vida más fácil, ya que la monitorización remota permite ahorrar tiempo y es más eficiente al usar menos recursos, mantiene la proactividad que se desea a la hora de gestionar una red y

mantiene un control constante. Monitorix es capaz de hacer esto monitorizando un número ilimitado de servidores remotos y crear grupos de estos servidores para tenerlos más organizados.

Por otro lado, la monitorización que lleva a cabo Monitorix a través de dispositivos LAN (*Local Area Network*) incluye la posibilidad de monitorizar todo tipo de dispositivos que estén asociados a dicha red – ordenadores, impresoras, … - y con ello, crear gráficos e informes que permitan un control y monitorización de dichas redes completo.

Alertas y emails

Monitorix, como toda herramienta de gestión de red que se precie, tiene entre sus principales criterios de funcionalidad la monitorización proactiva. Una de las maneras de mantener este tipo de control es a través de alertas y emails configurables que permiten al usuario anticipar de antemano problemas o amenazas que pongan en peligro la funcionalidad de los elementos que conforman la red.

Scripts y archivos de configuración adicionales

La posibilidad de crear scripts y nuevos archivos de configuración para que el usuario de Monitorix pueda recoger los datos que quiera es una de las características que posee esta herramienta. Esto puede ser muy útil a la hora de crear y monitorizar una red, pues será posible, gracias a la creación de dichos ficheros, poder recoger la información deseada.

2.4 Comparativa

Hasta aquí hemos profundizado en tres herramientas de gestión de red que podrían resultar útiles en el desarrollo del estudio. El gráfico siguiente muestra una comparativa entre las tres herramientas y las características que poseen cada una [Comparativa, 2015; Zabbix SIA, 2015; The Cacti Group Inc., 2004-2012; Sanfeliu, 2005-2015].

	Zabbix	**Cacti**	**Monitorix**
Open-source	Sí	Sí	Sí
Tipo de licencia	GPL v.2	GPL	GNU GPL v.2
SNMP soportado	Sí	Sí	Sí
Base de datos utilizada	Oracle MySQL PostgreSQL IBM DB2 SQLite	RRDtool MySQL	RRDTool

Sistemas operativos soportados	Linux IBM AIX FreeBSD NetBSD OpenBSD HP-UX Mac OS X Solaris: 9, 10, 11 Windows: 2000, Server 2003, XP, Vista, Server 2008, 7	Multiplataforma	Linux FreeBSD OpenBSD NetBSD
Plataformas	C PHP	PHP	Perl
Informes IP SLA	Sí	Sí	No
Agrupación lógica (users&groups)	Sí	Sí	No
Predicción de tendencias (capacidad de planificación)	Sí	Sí	Sí
Algoritmos de predicción soportados	No	Sí	Sí
Auto-descubrimiento	Sí	Con plugin	No
Soporte sin agente	Sí	Sí	No
Triggers/Alertas	Sí	Sí	Sí
Modificación de intervalos de recogida de datos	Sí	No	No
Basado en WebApp	Sí	Sí	Parcial
Monitorización en máquinas virtuales	Sí	Sí	Sí
Monitorización de Java app	Sí	No	No
Monitorización distribuida	Sí	Sí	Sí
Inventario	Sí	Sí	No
Control de acceso	Sí	Sí	Sí
Mapas de red	Sí	Con plugin	No
Uso de plantillas (templates&scripts)	Sí	Sí	Sí
Gráficos	Gráficos a tiempo real Opciones de visualización extensas	Características gráficas de RRDTool Visualización de gráficos en modo árbol jerárquico	Sí
Soporta IPv6	Sí	Sí	Sí

Última versión	2.4.5 (Abril 2015)	0.8.8f (Julio 2015)	3.8.0 (Septiembre 2015)
Otras características	Orientado a negocio y entornos complejos Monitorización de HW (temperatura, carga CPU…) Soporta *polling* y *trapping* Soporta SNMP *bulk request* Descubrimiento low-level	Puede mejorarse instalando Spine (aumenta rapidez de recogida de datos)	Opciones de gráficos muy extensas Recogida de estadísticas extensa

Tabla 1. Tabla comparativa de herramientas de gestión de red

LEYENDA
Sí
Parcial
No

Tabla 2. Leyenda de Tabla comparativa

2.4.1 Conclusiones de la comparativa

A partir de los datos obtenidos y del estudio realizado, se puede concluir que, de las tres herramientas utilizadas, Cacti y Zabbix son las más apropiadas para el despliegue deseado. Se descarta Monitorix, que aunque es una herramienta novedosa, no cumple con tantas expectativas y necesidades como sí lo hacen las otras dos.

A la hora de elegir finalmente entre una de las dos herramientas preseleccionadas, a priori Zabbix parece la más conveniente, ya que es una herramienta muy completa, que cumple las expectativas deseadas para el despliegue ya que tiene una magnífica proyección, no sólo a nivel básico sino también avanzado. Otra de las razones por las que se eligiría Zabbix, es su capacidad de representar gráficos a tiempo real y ser capaz de incorporar scripts personalizados para recoger datos concretos y después representarlos en gráficos detallados, a pesar de que Cacti también es capaz de realizar la personalización de scripts. Sin embargo, la razón fundamental de elegir Zabbix frente a Cacti sería la capacidad de la primera de modificar los intervalos de actualización y recogida de datos, algo que en nuestro despliegue tiene que ser del orden de segundos, mientras que Cacti tiene un intervalo mínimo de un minuto, intervalo demasiado alto para el estudio que se está llevando a cabo.

No obstante, se probarán ambas herramientas en un despliegue preliminar, para ver in situ las ventajas y/o desventajas que tiene una herramienta frente a otra y viceversa, y así poder realizar un estudio completo de las dos herramientas de gestión de red seleccionadas.

2.5 Requisitos y restricciones

Para llevar a cabo tanto los despliegues de prueba como los finales, deberemos de disponer de:

- Terminal PC, con entorno Linux (Distribución Debian)
- Raspberry Pi, con distribución Raspbian (Se usará para las pruebas de validación)
- Router UAV
    - Fuente de alimentación router UAV
    - Ratón (adaptador PS2 y USB)
    - Teclado (adaptador PS2 y USB)
    - Pantalla para el router UAV
- *Switch*
- Cables de red RJ45
- Cable HDMI para conexión de la Raspberry Pi a pantalla (Se usará para las pruebas de validación)
- Conexión a Internet
- Gestor de máquinas virtuales (Se usará para las pruebas de validación)
- Disponibilidad de los laboratorios de Telemática
- Paquetes de las herramientas elegidas (instaladas con comandos apt-get [Noronha Silva & Mora, 2003; Ubuntu, 2015])

2.6 Marco regulador

Este desarrollo como se ha mencionado con anterioridad, está basado en el manejo de los vehículos aéreos no tripulados, así que está sujeto a marcos regulatorios pertinentes. Además, como el estudio que se hace en este Trabajo de Fin de Grado es sobre las herramientas y el despliegue de la gestión de red en dicho vehículo aéreo no tripulado, el marco regulatorio sobre el que se va a basar va estar limitado a este estudio.

En primer lugar, se acogerá a la Ley Orgánica 15/1999 de 13 de diciembre de ***Protección de Datos de Carácter Personal, (LOPD)*** [LOPD, 1999], ya que en el supuesto y potencial caso que se quiera crear un usuario en Zabbix para monitorizar sus propios dispositivos, se habrán de añadir datos de carácter personal – tales como nombres, direcciones de correo electrónico y contraseñas.

Además, se ha de tener en cuenta la ***Ley General de Telecomunicaciones***, la Ley 9/2014 [LGT, 2014], que engloba la creación de un marco pertinente para las telecomunicaciones en España para facilitar las inversiones necesarias para el desarrollo de esta economía digital, eliminando barreras y fomentando la competitividad y la protección del usuario. Una de las novedades de esta ley tendrá como objetivo principal facilitar el despliegue de redes de nueva generación, fijas y móviles, para ampliar su cobertura. Como vemos, este marco es muy importante dentro del estudio que se está llevando a cabo, pues implica a las redes creadas por los UAV y la gestión de las mismas.

En relación al uso del vehículo aéreo no tripulado [Ley, 2016], es necesario acogerse al **Real Decreto-ley 8/2014**, del 4 de julio [Ley Drones, 2014], de aprobación de medidas urgentes

para el crecimiento, la competitividad y la eficiencia. En este decreto se trata el uso civil y comercial de aeronaves controladas remotamente. Junto a este decreto es necesario acatar también el régimen general de la **Ley 48/1960** sobre ***Navegación Aérea*** [Ley Navegación Aérea, 1960], pues a pesar de que no se va a realizar ningún vuelo, el UAV debe estar sujeto a las restricciones y normas establecidas en esta Ley.

Capítulo 3

Diseño y desarrollo de la solución

3.1 Introducción

En este apartado cabe destacar que es importante discernir entre diseño y desarrollo. En primer lugar, se llevará a cabo una evaluación práctica preliminar que consistirá en la aplicación de las dos herramientas de monitorización de red elegidas previamente – Cacti y Zabbix – con sus correspondientes particularidades. Al final de dicha evaulación, se elegirá de entre las dos finalmente cuál será la que se elija para el diseño y desarrollo de la solución con su implícita implementación.

En el apartado de diseño se va a realizar un despliegue sobre un equipo dedicado, obteniendo datos a través del protocolo SNMP y scripts personalizados, para después crear *screens* con los gráficos de dichos datos obtenidos. Posteriormente, en el desarrollo de la solución, como se ha mencionado, se aplicará la herramienta elegida con los parámetros a monitorizar previamente elegidos y se verá su despliegue de forma gráfica para comprobar su validez y efectividad.

3.2 Despliegue preliminar

3.2.1 Introducción

En un primer lugar, se va a realizar un pequeño despliegue de pruebas en el que se contará con un ordenador que actúe como cliente, un *switch* que interconecte al cliente con el router, y el router UAV (con una distribución Debian), que es el router que va colocado en el sistema aéreo no tripulado, y el cuál será el que habrá que monitorizar en este despliegue para generar las gráficas de estadísticas que podremos conseguir tanto con Zabbix como con Cacti. En este despliegue se ha utilizado además un equipo dedicado con distribución Ubuntu 10.04 y otros componentes como cables RJ-45 para interconectar los distintos útiles. Se puede además

consultar el esquema del despliegue preliminar y la configuración previa en el apartado de Anexos (Anexo I y II).

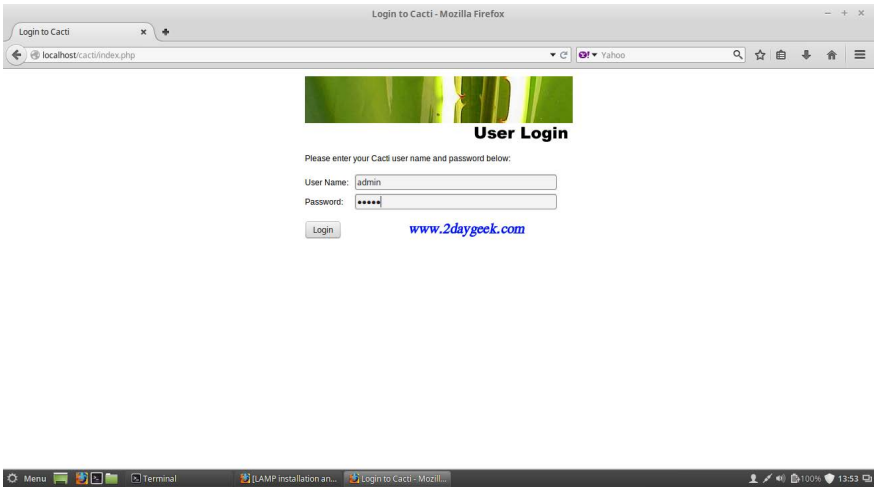
También se van a explicar los pasos para la instalación de las dos herramientas elegidas con pasos análogos ya que se utilizará la herramienta **apt-get** de la distribución Debian para descargar e instalar dichas herramientas en ambos dispositivos. Este apartado puede ser consultado en el apartado de Anexos (Anexo III).

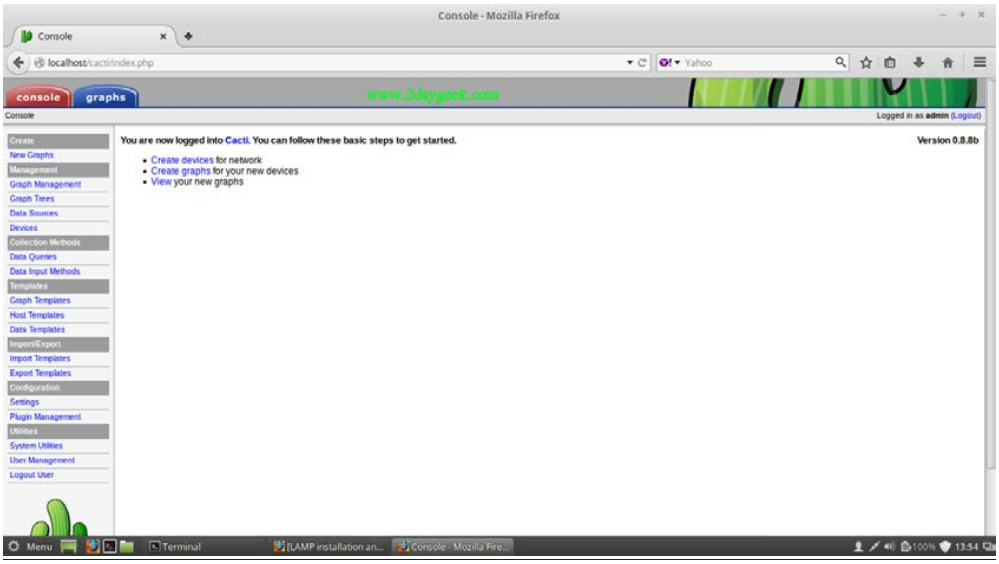
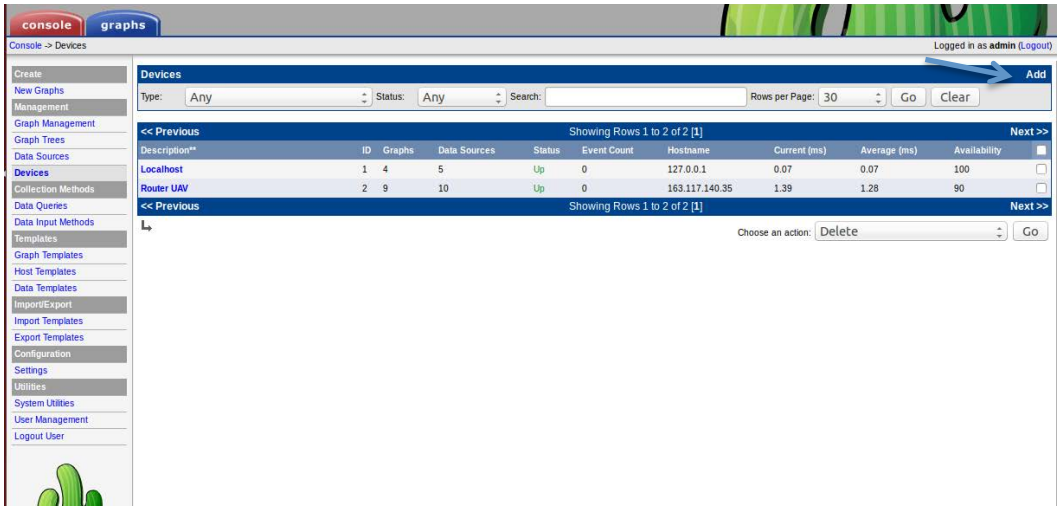
Este primer despliegue servirá como práctica para entender y familiarizarse con las nuevas herramientas de gestión de red, así como para ver in situ sus ventajas y desventajas, respectivamente, y finalmente decantarse por una de ellas.

3.2.2 Información obtenida con las herramientas en el despliegue preliminar

3.2.2.1 Cacti

A continuación se van a observar algunas imágenes capturadas de estadísticas a partir de plantillas ofrecidas por defecto por Cacti tanto en el sistema cliente como en el router UAV. El procedimiento para monitorizar los dispositivos es sencillo. Únicamente hay que seguir los pasos explicados en el apartado de creación de gráficos en la sección del estado del arte de Cacti.

Paso	Descripción del paso
1	Se accede a Cacti con un paso análogo al paso número 11 de la tabla del despliegue preliminar de la red (ver Anexo III, tabla 15), aunque aquí redirigirá directamente a la página de registro de usuario.  **Figura 17. Página de inicio de Cacti** Las credenciales son las siguientes: 1. User Name: *admin* 2. Password: *1234*

2	La primera pantalla que aparece es la de inicio de Cacti.  **Figura 18. Pantalla de inicio de Cacti**
3	Una vez dentro, se puede observar la columna de navegación en la que se pueden crear dispositivos que administrar y crear los gráficos pertinentes para cada dispositivo.
	Para la creación de un dispositivo que se desee monitorizar, se ha de pulsar en la pestaña **Devices.** Ahí aparecerán todos los dispositivos que están gestionados hasta ese momento por Cacti. En la esquina superior derecha hay un botón, **Add**. Este sirve para el propósito actual, crear un nuevo host para monitorizar.  **Figura 19. Pantalla *Devices* para la creación de un host en Cacti** Cuando se pulsa en ese botón, se puede introducir toda la información pertinente para el nuevo *host* creado. Por ejemplo, para la creación de router UAV, se incluyeron los siguientes datos.

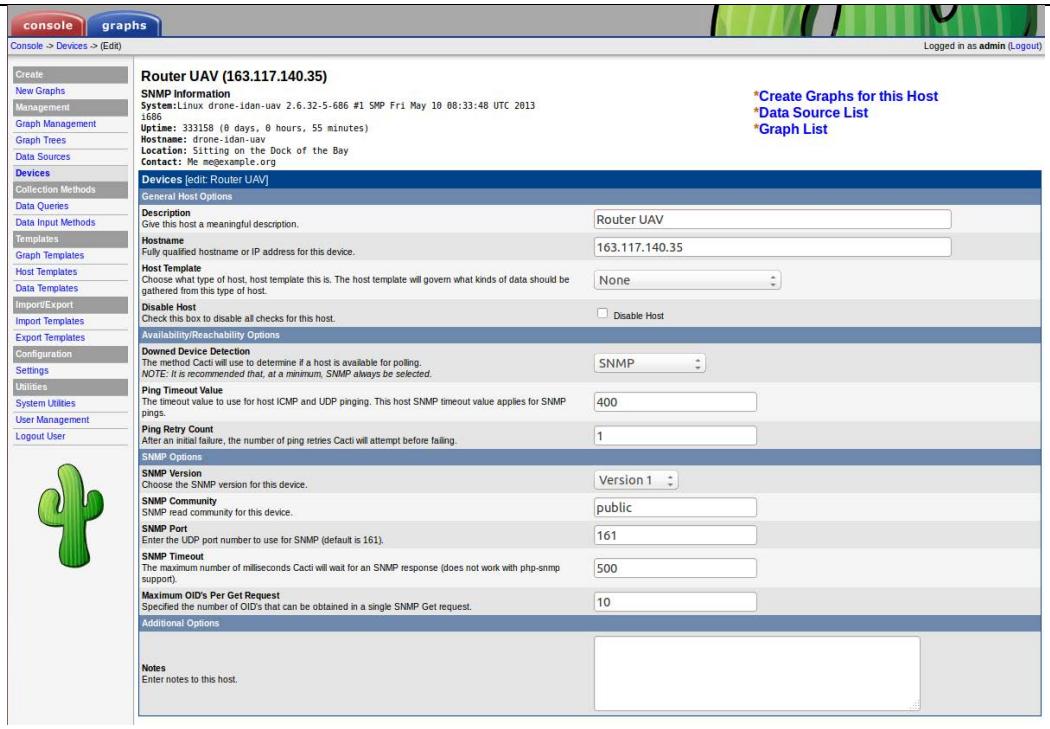


Figura 20. Información de *host* creado en Cacti

5 La pantalla de dispositivos manejados tiene la siguiente apariencia, pudiendo modificar datos y parámetros de manera sencilla.

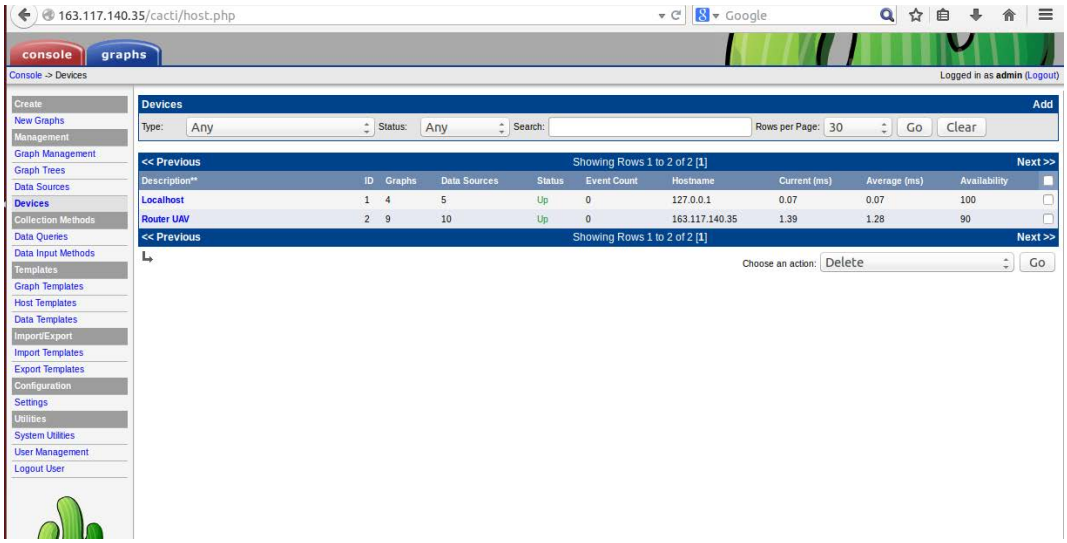


Figura 21. Dispositivos manejados por Cacti

6 Para crear gráficos, hay que ir a la pestaña de la columna de navegación **New Graph**. Ahí se elige el host para el que crear gráficos y el tipo de gráficos que se desean representar:

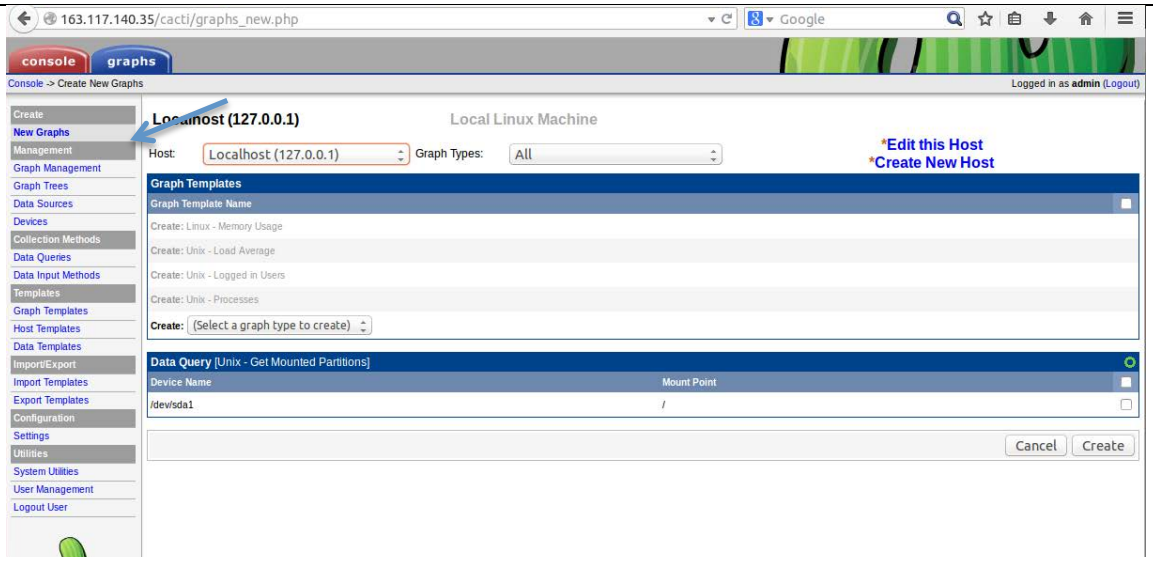


Figura 22. Creación de gráficos en Cacti

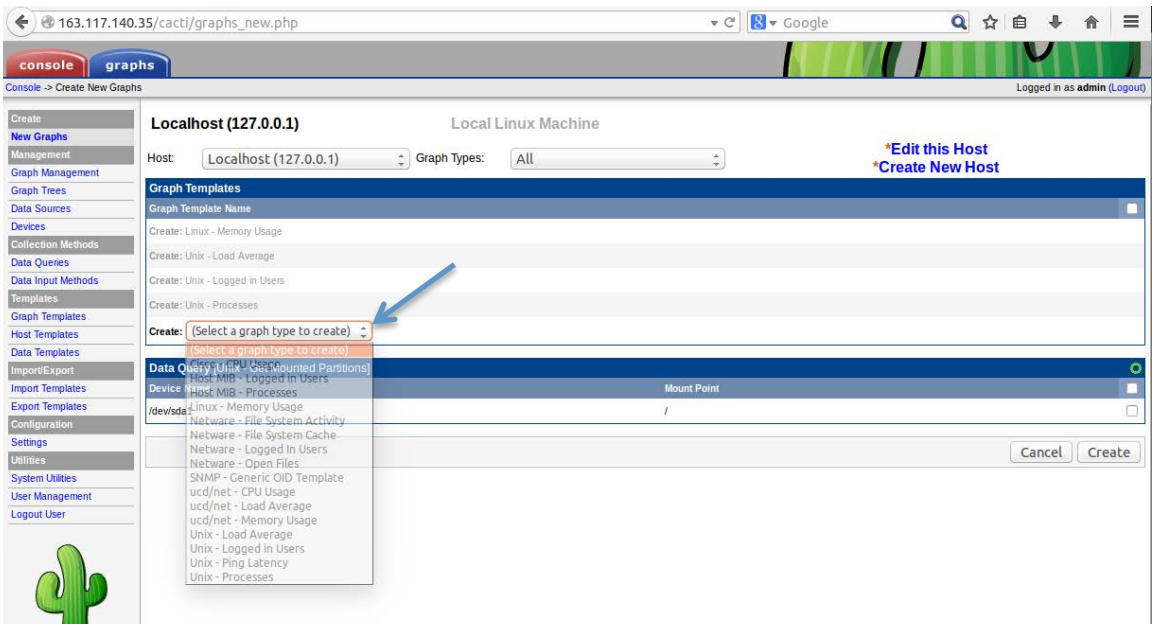


Figura 23. Elección de tipo de gráfica en Cacti

7	También se puede ver la información de los gráficos que existen y su información en el modo **Graph Trees**:

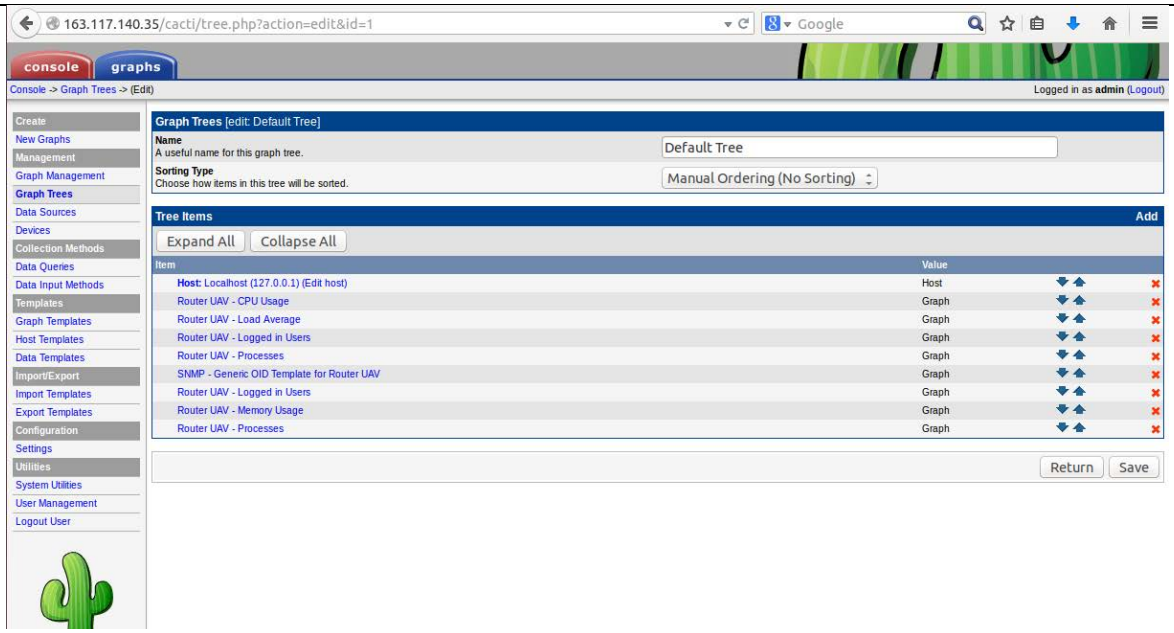


Figura 24. Modo *Graph Tree* de Cacti

En esta vista, se pueden ordenar los gráficos y obtener información de éstos para luego visualizarlos en el árbol de gráficos.

8 En el apartado **Graph Management** se puede elegir los gráficos y los árboles que queremos mostrar de forma personalizada.

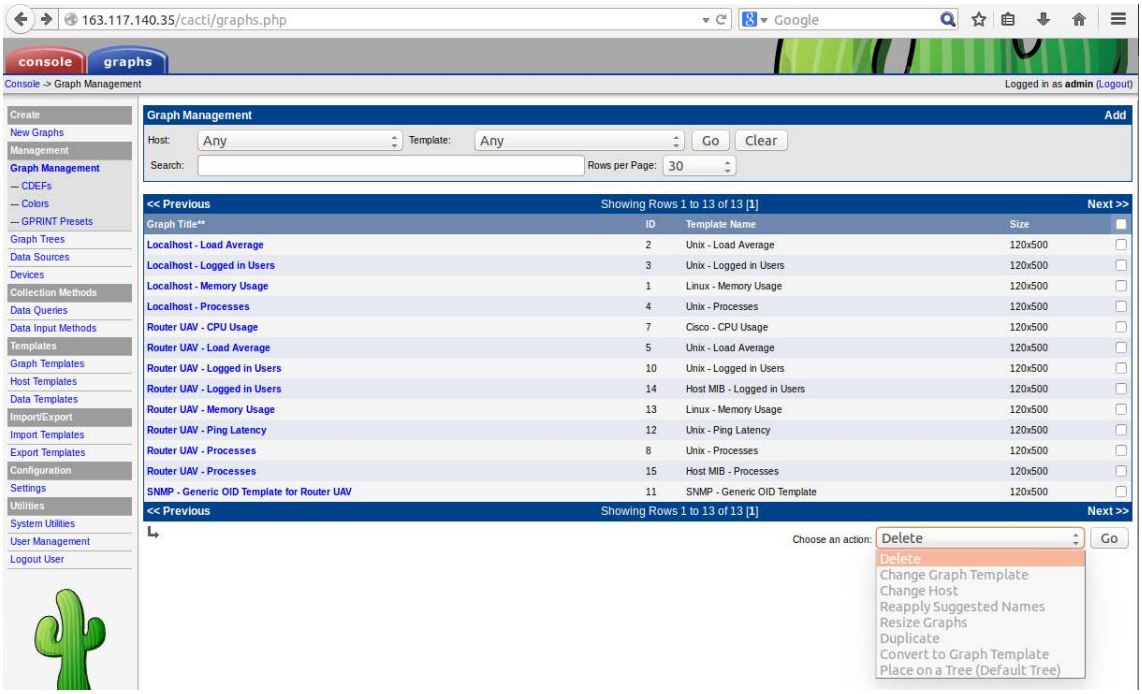


Figura 25. *Graph Management* en Cacti

9 Por último, otro apartado muy importante para este despliegue son los *templates* o plantillas. Los *templates* son las plantillas que usa Cacti para obtener datos estadísticos de distintos dispositivos y representarlos después en gráficas. Los hay de tres tipos: *host*, *graph* y *data sources*, y cada uno

funcionan como plantilla para dispositivos, gráficos y fuentes de datos, respectivamente.

Hay varios por defecto, que son los que se usarán en las siguientes imágenes, pero también se pueden exportar e importar scripts personalizados que actúen como **templates**. Por ejemplo, si se quiere que Cacti represente una gráfica con datos de la temperatura de la CPU de un dispositivo, será neceario crear un script que funcione como plantilla y recoja esos datos, para así después representarlos.

Figura 26. Plantillas de gráficos en Cacti

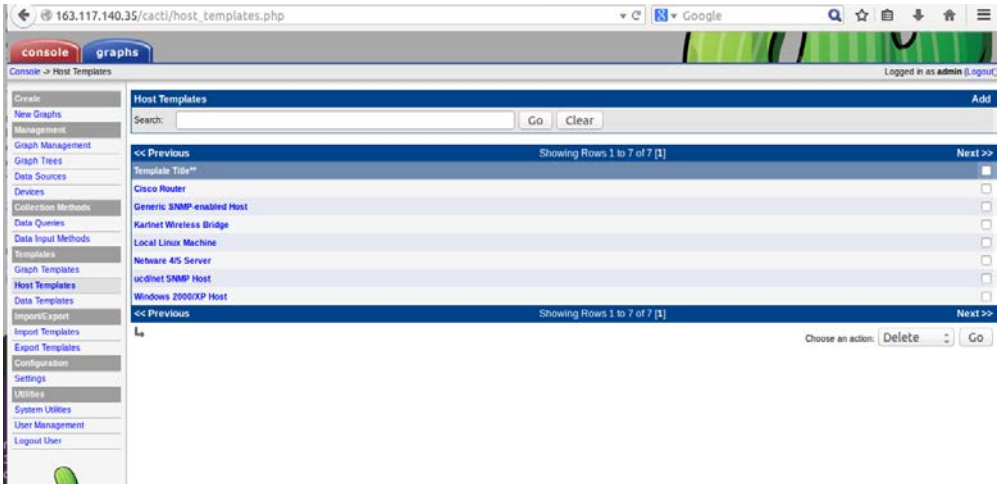


Figura 27. Plantilla de dispositivos en Cacti

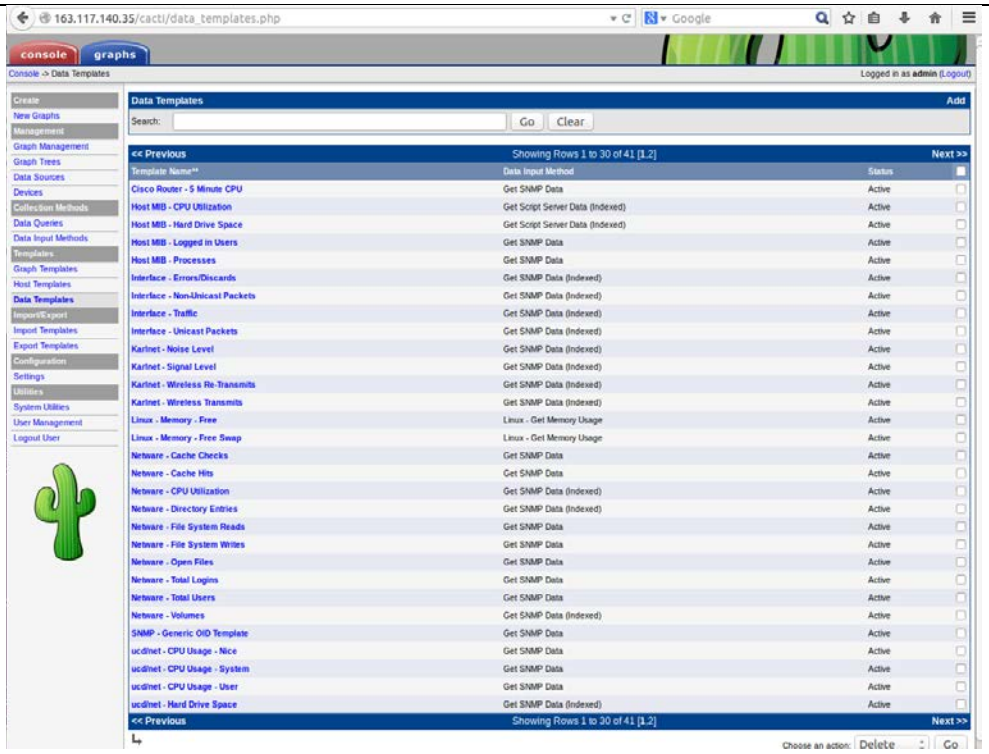
	 **Figura 28. Plantillas de fuentes de datos en Cacti**
10	Hasta aquí llegan los pasos fundamentales para crear dispositivos y gráficas en Cacti, pero como ya se ha mencionado con anterioridad, esta herramienta ofrece multitud de posibilidades para la monitorización de red y de sus componentes.

Tabla 3. Utilización de Cacti

Gráficos

Cliente (Localhost)

En este apartado se van a mostrar algunas gráficas de los datos obtenidos con Cacti para el terminal cliente. En un principio no se iban a obtener gráficas de dicho terminal, pero para probar con la nueva herramienta Cacti y familiarizarse un poco con ella, se decidió probarla en el cliente y ver cómo recogía las estadísticas y después las representaba.

Como se puede observar en las figuras 29, 30, 31 y 32, se recogen gráficos de cuatro tipo de estadísticas en el equipo llamado **Localhost** (en este caso son estadísticas del PC, donde se ha instalado Cacti), que son:

3. Uso de memoria
4. Carga media de CPU
5. Usuarios registrados
6. Procesos activos

Las cuatro gráficas son *templates* de datos por defecto de Cacti y se representan en el periodo de una hora, aunque este periodo puede ser elegido, entre ser personalizado e intervalos de media hora hasta dos años. Además, cada gráfico tiene sus propias características, dibujando

por colores las distintas propiedades que pueden tener cada estadística. Por ejemplo, en el de uso de memoria, se pinta la memoria libre como naranja oscuro y la memoria ocupada o *swap*, como amarillo oscuro. También se recogen datos de medias, máximos y datos actuales. Estos modos de representación también pueden ser personalizados en la consola de Cacti.

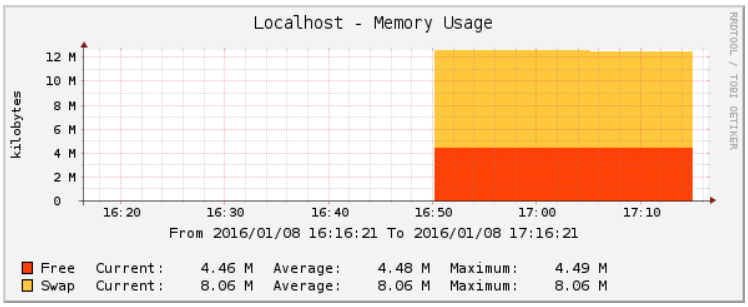


Figura 29. Uso de memoria en PC en Cacti

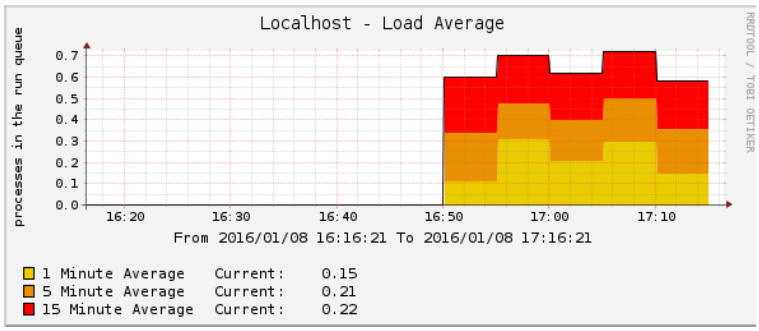


Figura 30. Carga media en PC en Cacti

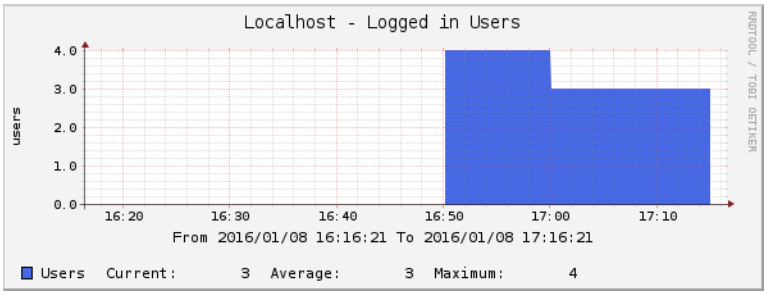


Figura 31. Usuarios registrados en PC en Cacti

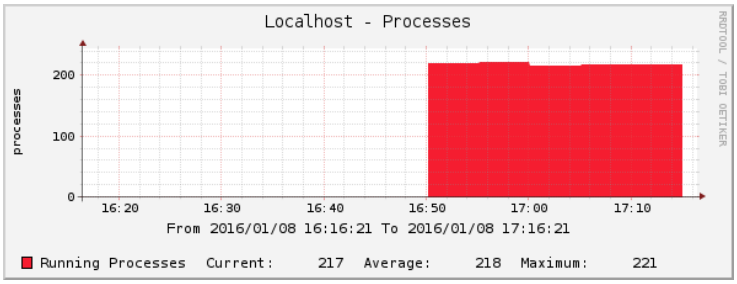


Figura 32. Procesos en PC en Cacti

Router UAV

Al igual que con el terminal cliente, se recogen las mismas gráficas (pero con un periodo menor de representación de media hora) para el router UAV previamente configurado con sus parámetros pertinentes, y estos son los resultados obtenidos (figuras 33, 34, 35 y 36):

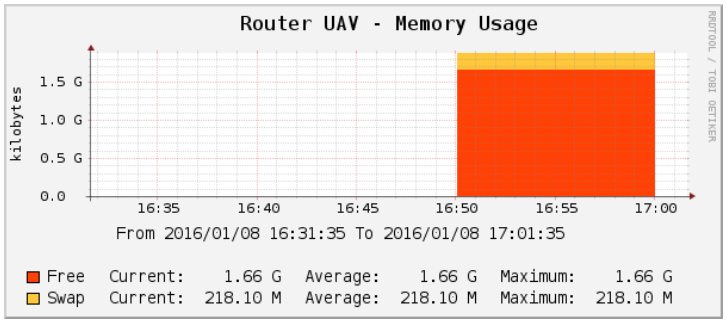


Figura 33. Uso de memoria en router UAV en Cacti

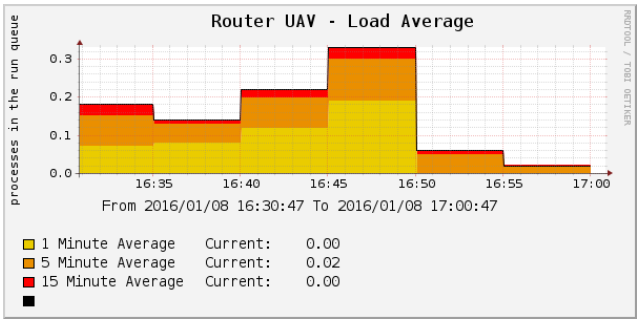


Figura 34. Carga media en router UAV en Cacti

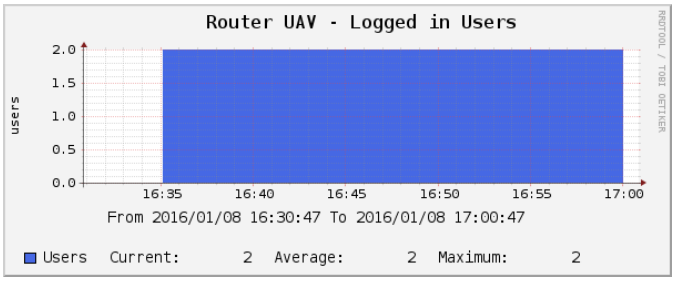


Figura 35. Usuarios registrados en router UAV en Cacti

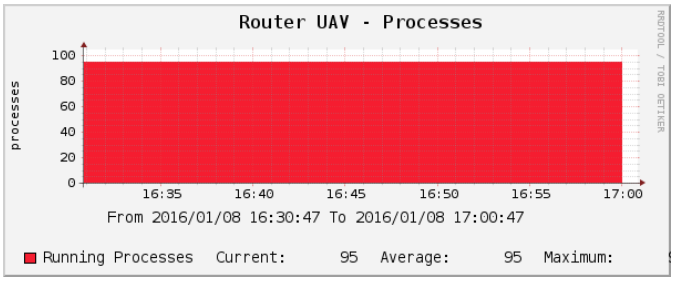


Figura 36. Procesos en router UAV en Cacti

Como se puede observar, la representación de datos estadísticos en este caso es cada cinco minutos y este es el periodo mínimo de recogida para Cacti. Tristemente, esto no sirve para los propósitos porque en el escenario en el que hay que poner en práctica estas herramientas de monitorización y gestión de la red es un escenario en el que se necesitan recoger muestras de datos en periodos de tiempos muy cortos. Esto es debido a que la duración de los vuelos es limitada, por ello, conviene recoger muestras de algunos parámetros con frecuencia más alta, para tener más datos durante el vuelo e incluso poder reaccionar a problemas en dicho trayecto (detectando los problemas de forma rápida).

Por ejemplo, se puede imaginar un vuelo no tripulado de un dispositivo UAV de una hora. Si se recogen muestras de información en periodos de cinco minutos solo se obtendrían 12 muestras. Esto no permitiría crear gráficos óptimos o bien visibles. Por esa razón es importante el estudio y familiarización de otra herramienta como es Zabbix, para sopesar los pros y los contras y ver cuál es más eficaz en el despliegue deseado.

3.2.2.2 Zabbix

En este apartado se va a acceder a la consola de Zabbix a través de la aplicación Web [Zabbix, 2016] y a partir de ahí, se van a explicar los pasos que hay que seguir para crear un dispositivo a monitorizar y cómo se generan las gráficas de dichos dispositivos.

Paso	Descripción del paso
1	Se accede a la consola de Zabbix escribiendo en el navegador http://direcciónIP/zabbix, en el caso de este despliegue, http://10.1.2.1/zabbix (10.1.2.1 es la dirección IP del router UAV en la interfaz eth0) que redirige a la página de configuración inicial de la consola de Zabbix, http://10.1.2.1/zabbix/setup.php

Figura 60. Configuración de la consola de Zabbix I

Se pulsa a *Next*, y se siguen los siguientes pasos:

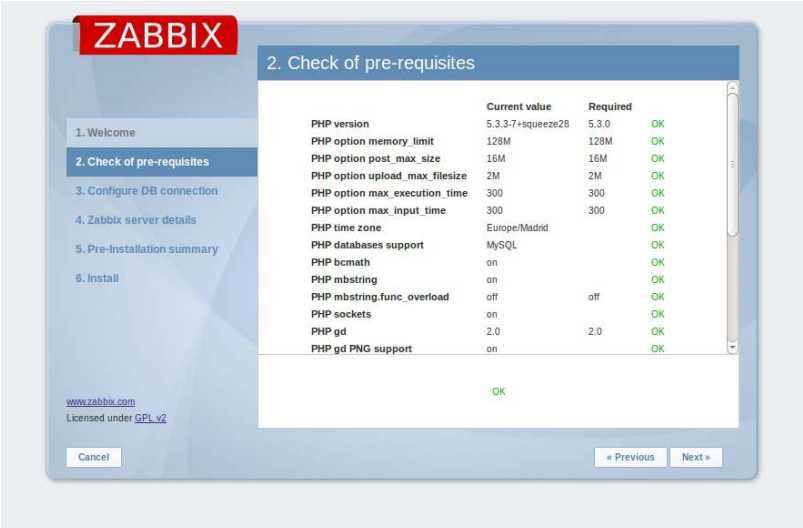


Figura 61. Configuración de la consola de Zabbix II

En el chequeo de los prerrequisitos todo tiene que estar en verde para continuar, como se puede apreciar en la imagen. Si no estuviese en verde, habría que modificar el archivo de configuración de Zabbix para que todo estuviese correcto. Principalmente, los errores que suelen dar en esta pantalla es por no haber configurado correctamente la zona horaria en el archivo de configuración PHP.

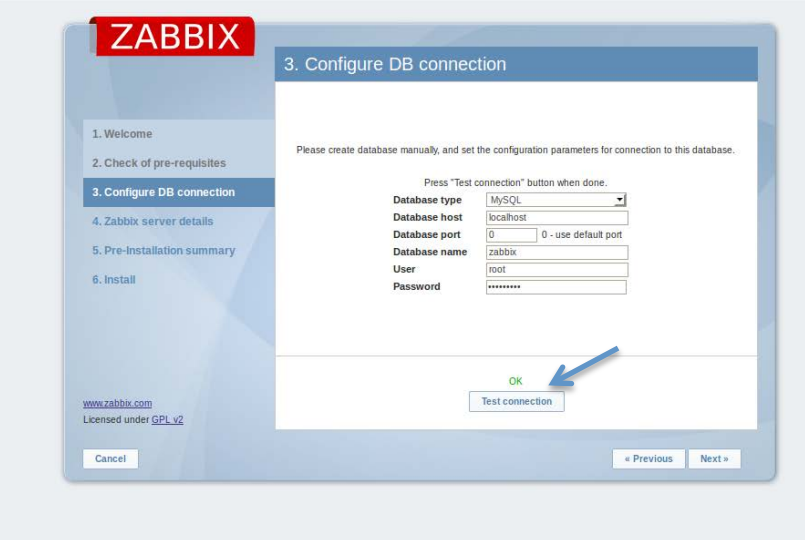


Figura 62. Configuración de la consola de Zabbix III

Ahora hay que introducir la contraseña administrativa de MySQL (*root* con la contraseña que se dio en la instalación) y se pulsa a *Test connection*. Así, si todo va bien, se puede seguir y dar a *Next*. Igual que antes, si algo fuese mal en este paso habría que mirar el error y corregirlo configurando correctamente el archivo de configuración de Zabbix.

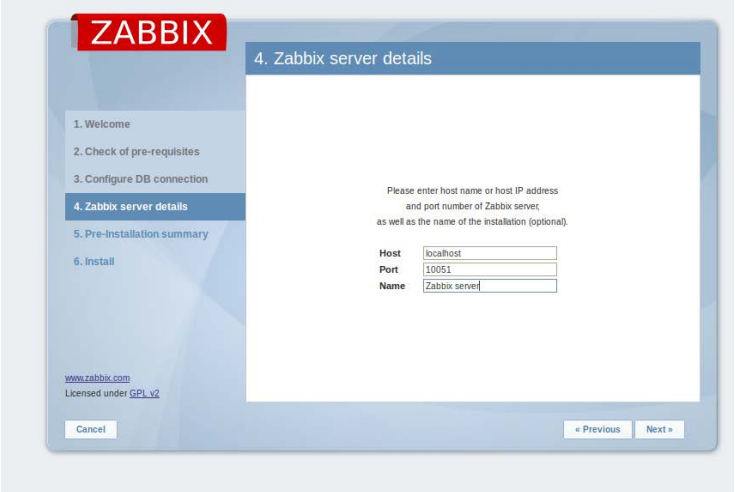


Figura 63. Configuración de la consola de Zabbix IV

Se pulsa en *Next*.

Figura 64. Configuración de la consola de Zabbix V

Se observa que está todo correcto en la figura anterior y se continua dando a *Next*.

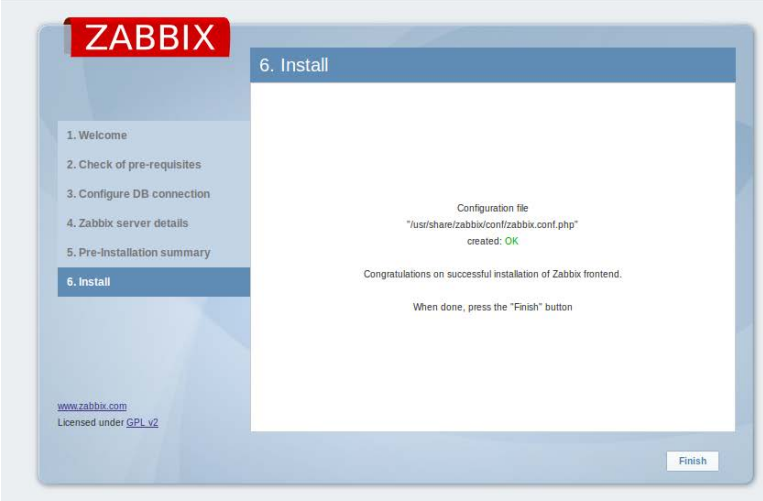
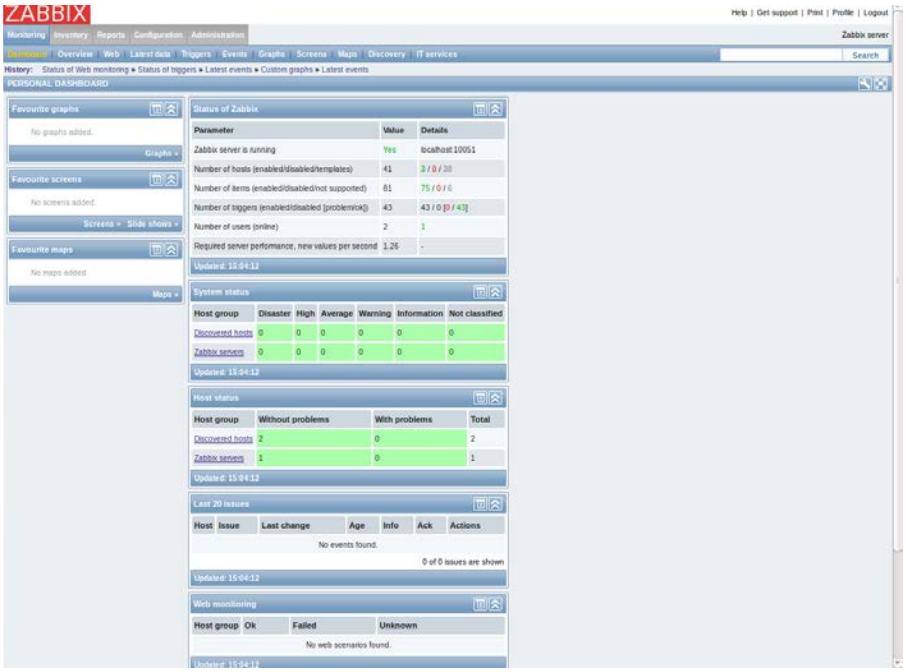
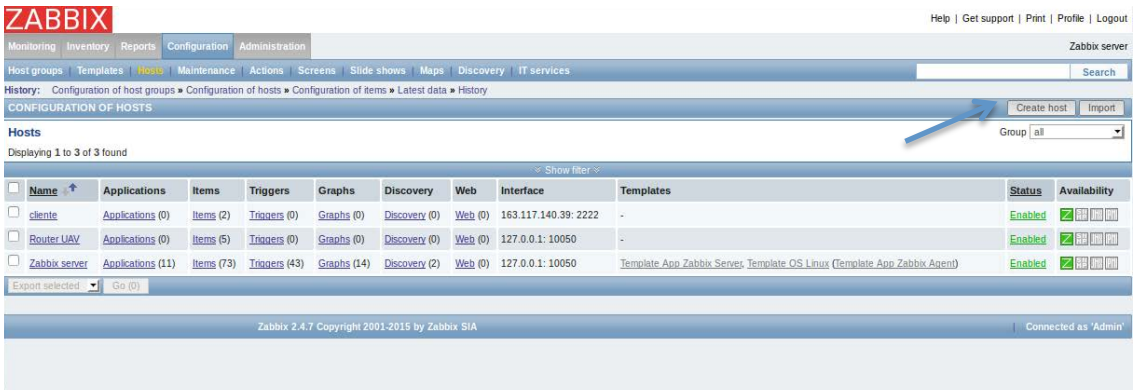


Figura 65. Configuración de la consola de Zabbix VI

Una vez llegamos a esta pantalla, se pulsa en *Finish* para poder finalizar la configuración de la consola Zabbix.

	Figura 66. Configuración de la consola de Zabbix VII Finalmente, ya se puede acceder a ella con unas credenciales. Pueden crearse o acceder como usuario administrador, cuyo nombre de usuario es *admin* y su contraseña, *zabbix*.
2	Cuando se accede, se observa la vista principal de la consola, donde se puede acceder a multitud de menús y características:  **Figura 67. Vista principal de la consola de Zabbix**
3	Para que Zabbix pueda reconocer el dispositivo a monitorizar, en primer lugar, hay que crearlo. Para ello, en la vista principal se accede a **Configuration** y posteriormente a **Hosts**. Esta es la vista que se obtiene:  **Figura 68. Vista Hosts de Zabbix** Si se quiere crear un nuevo *host*, lo único que hay que hacer es dirigirse a la esquina superior derecha donde pone *Create host* y clicar. La vista Creación de Host es la siguiente:

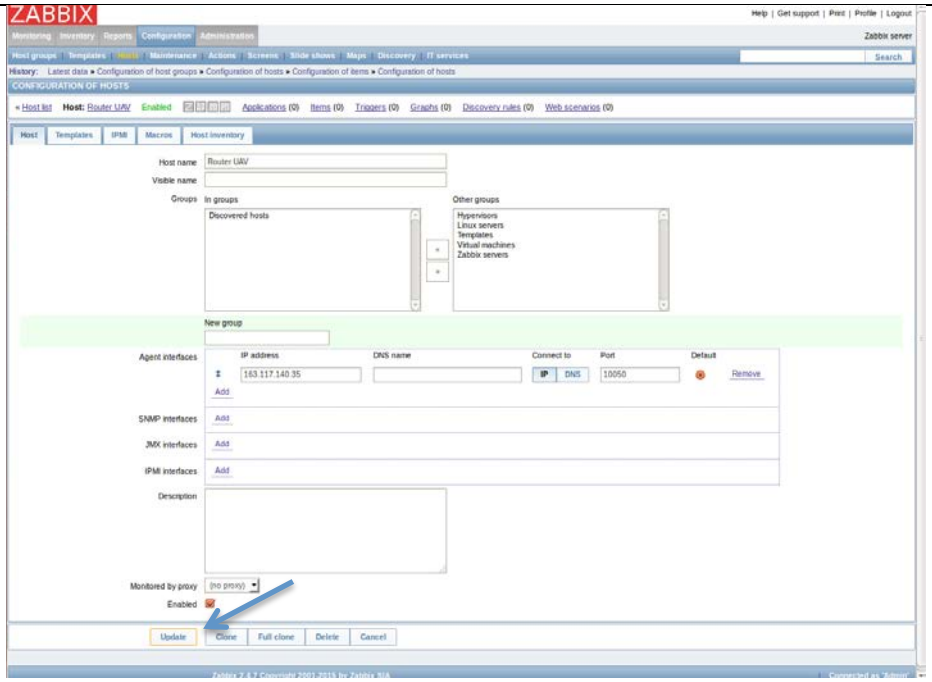


Figura 69. Vista creación de Host

Aquí se puede dar un nombre al *host*, configurar al grupo o grupos al que pertenezca, asignarle la dirección IP y otros parámetros más. Una vez introducidos todos los datos como se ve en la imagen anterior, hay que clicar en *Update* para crearlo.

Para que el nuevo host esté disponible, será necesario que una vez en la vista de **Hosts** se pulse en el **Estado** y cambiar de *Disabled* a *Enabled*.

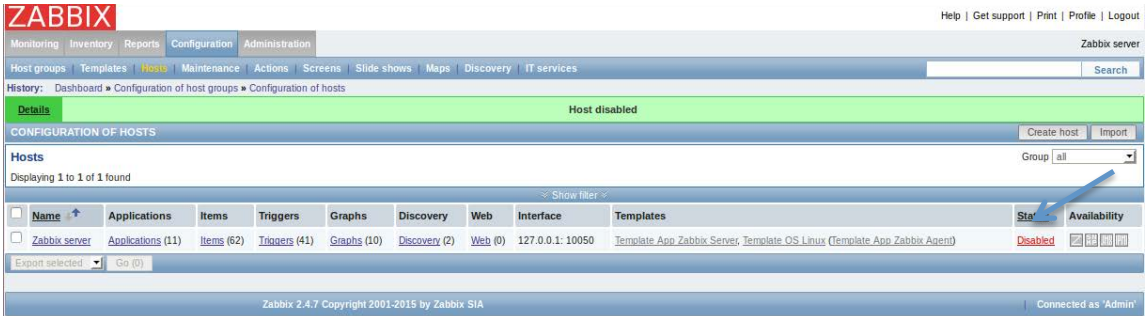


Figura 70. Disabled Host en Zabbix

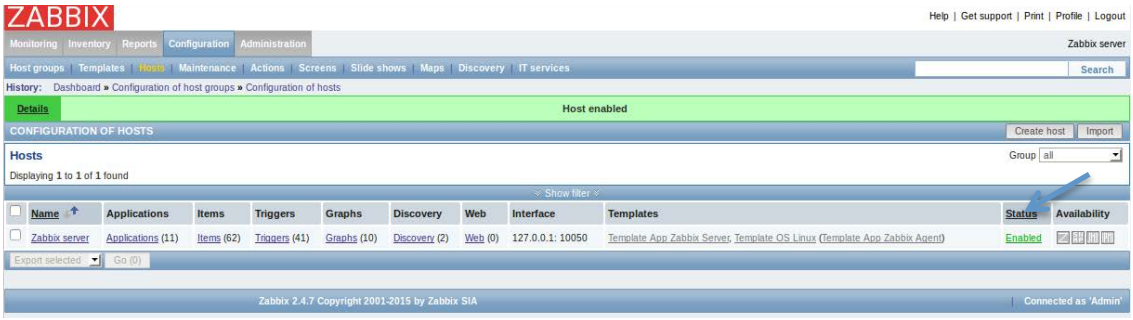


Figura 71. Enabled Host en Zabbix

NOTA: En estas figuras se realizó el ejemplo de imágenes con el mismo servidor Zabbix.

4

Una vez creado el host que se quiere monitorizar, hay que de crear *Items*. Los *items* son las estadísticas y datos que queremos recoger de dicho dispositivo monitorizado. En el siguiente ejemplo se va a observar cómo crear un item para el host creado con anterioridad, Router UAV.

Figura 72. Creación de item para host en Zabbix

Para crearlos, es neceario pulsar en la pestaña **Hosts** y elegir uno de los *hosts* existentes. Allí, en la esquina superior derecha, existe un botón llamado *Create new item*. Se pulsa y se introduce la información y los parámetros necesarios para la creación del *item*.

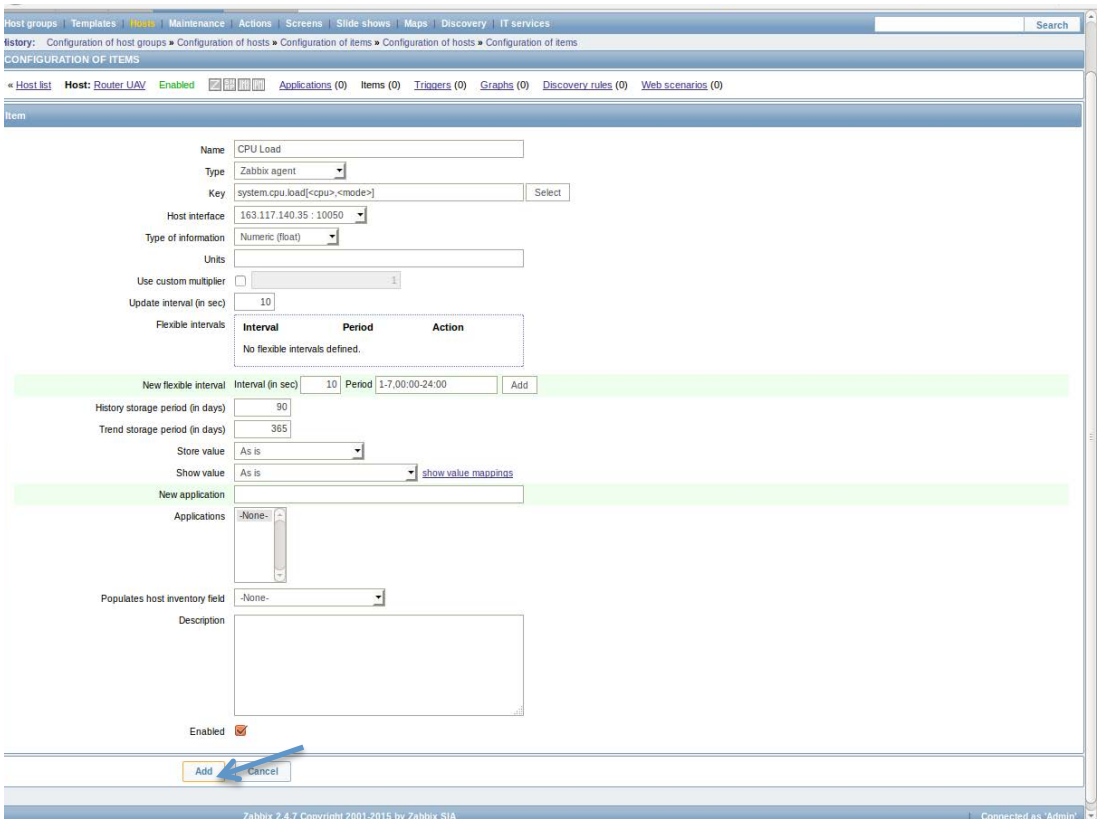
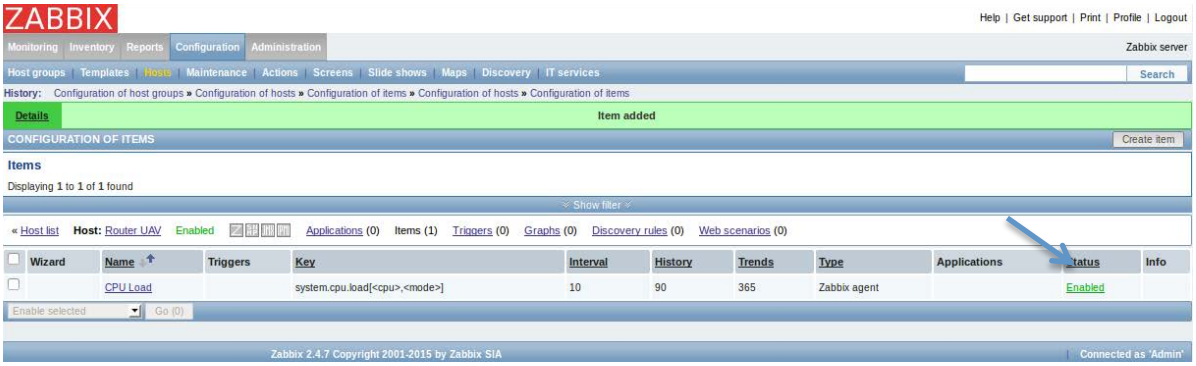


Figura 73. Pantalla de creación de item en Zabbix

Una vez introducidos los parámetros pertinentes, que son esencialmente: *name*, *key* y *type of information* (el resto viene por defecto), se puede proceder a crear el item pulsando *Add*.

En el caso que ocupa, por ejemplo, se necesitaba que cogiese datos de la carga de la CPU. Se incluye

	el nombre, la *key* (de forma genérica, que luego se puede modificar con los parámetros que están entre corchetes) y el tipo de información, y se pulsa *Add*. Así, ya está creado el item para el host.  **Figura 74. Activación de item para *host* en Zabbix** Por último, es necesario que como los hosts, los items se activen. Esto se hace una vez en la vista de items, en **Status**. Se pulsa y pasa de *Disabled* a *Enabled*. Ya está listo el item listo para recoger datos y después representarlos en gráficas.
5	Para ver las gráficas creadas para cada host hay que ir a la vista **Monitoring**, y una vez ahí pulsar **Latest Data**.  **Figura 75. Vista *Latest Data* sin filtrado** Ahí a través de unos filtros se puede elegir que *hosts* visualizar.

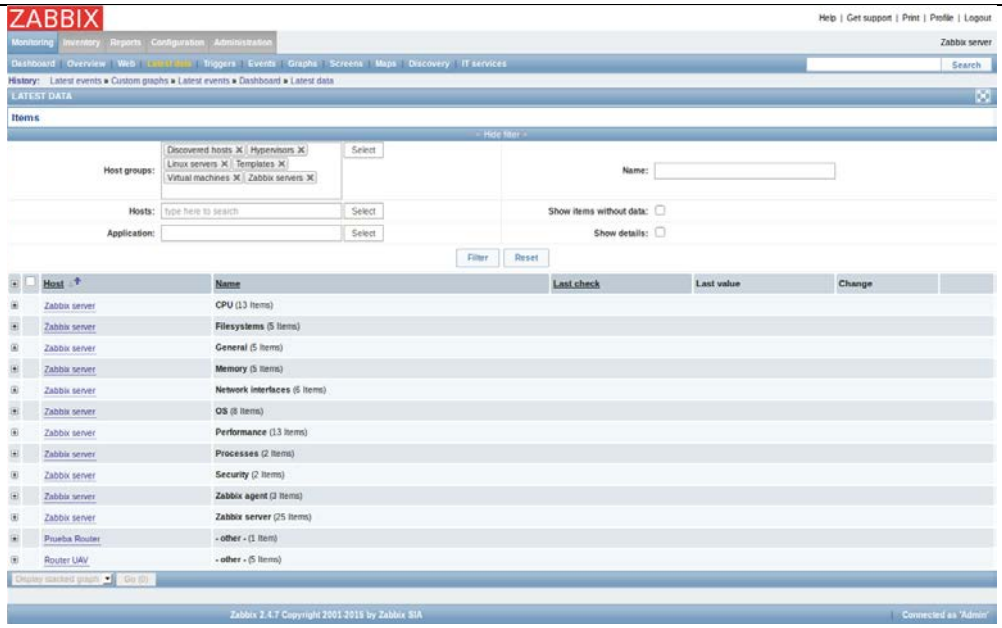


Figura 76. Vista *Latest Data* con filtrado

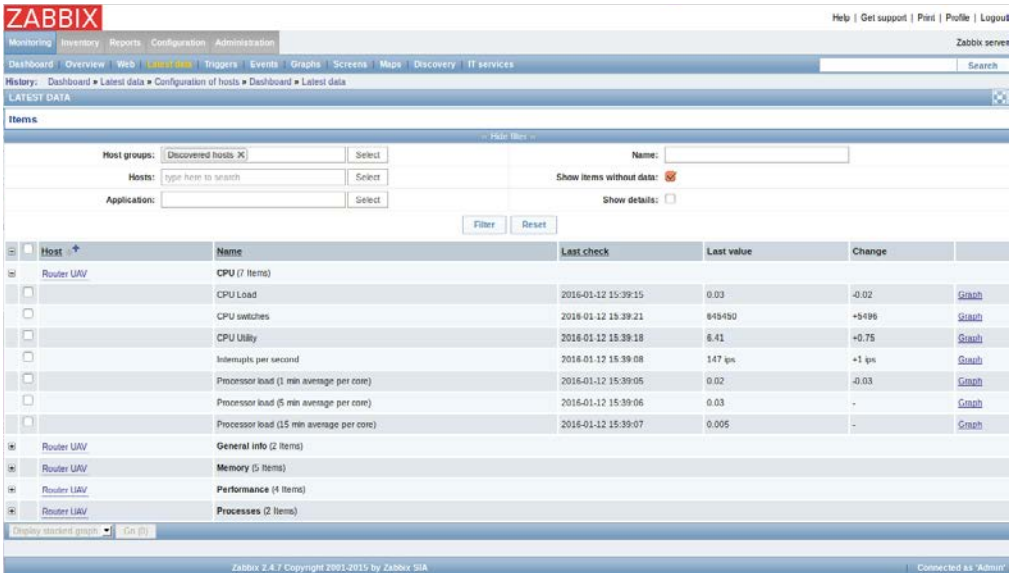


Figura 77. Vista *Latest Data* con filtrado para Router UAV

Una vez aparecen los *hosts* y las gráficas que hay disponibles, se da a un botón que pone *Graph* o *History* (según el tipo de gráfico o representación que ofrezca el item) y se puede visualizar el gráfico. A menudo, cuando se trata de la creación de un nuevo *item* para un nuevo *host*, la visualización de los gráficos puede tardar un poco. Sólo hay que esperar aproximadamente 60 segundos para tener los primeros datos y que estos se representen en la gráfica.

6

Esta es la funcionalidad básica de Zabbix. Tiene muchísimas más posibilidades para explorar y crear gráficos, entre otras cosas. Al crear un host para que sea monitorizado se tiene la posibilidad de hacer una monitorización bastante completa gracias a que se pueden crear, entre otros:

- **Aplicaciones**: Grupos de items para ordenarlos en el dashboard.

	• **Items**: Como se ha mencionado, es la información y/o datos que se recogen del host para representarse. • **Triggers**: También conocidos como Alarmas. Permiten poner umbrales de tiempo a los items para monitorizarlos de la forma deseada y proactivamente. • **Graphs**: Gráficos personalizados de items. • **Discovery Rules**: Reglas de descubrimiento de los hosts. • ***Screens***: Conjunto de gráficos en una misma pantalla. También se ha podido apreciar que Zabbix es una herramienta de monitorización bastante más completa que Cacti, pues como se ha mencionado antes, tiene muchas más características y posibilidades, así como más plantillas y tipos de gráficos (gráficos circulares, de barras, conjuntos, …) y opciones de configuración e informes más completos.

Tabla 4. Utilización de Zabbix

Gráficos

Al igual que con Cacti, se decidió probar Zabbix en un despliegue inicial sencillo para familiarizarse con la herramienta y sus funcionalidades. Como se pudo observar, Zabbix era más complejo y tenía muchas más posibilidades a la hora de representar gráficos y obtener datos que Cacti. Una de las características que la hicieron más óptima a la hora de elegir frente a Cacti fue la posibilidad de modificación de intervalos de representación en gráficas, para variar y elegir periodos más pequeños – del orden de segundos – para hacerlo más razonable con respecto al despliegue.

Esta característica sin embargo, se probará en el apartado Desarrollo de la solución. De momento, en las imágenes de este apartado se trabajará con el intervalo por defecto, que es un minuto.

A continuación, se observarán algunas gráficas, tanto simples como personalizadas, tanto del servidor Zabbix – cuyos items fueron creados a partir de templates por defecto al momento de la instalación de la herramienta - como otro *host*, llamado router UAV, que se creó para practicar la creación de *hosts* e *items*.

Zabbix Server

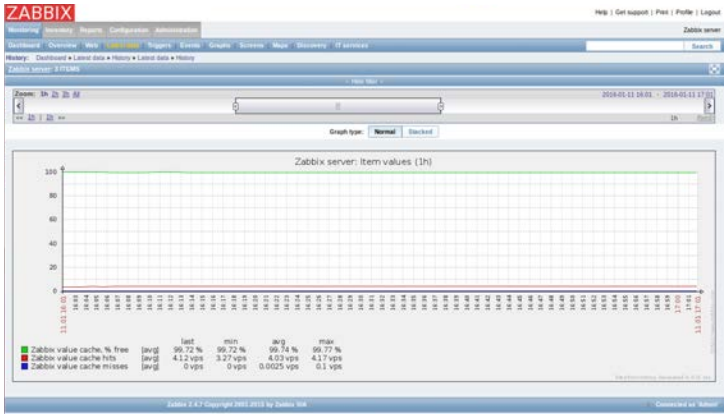


Figura 37. Valores de caché en Zabbix Server

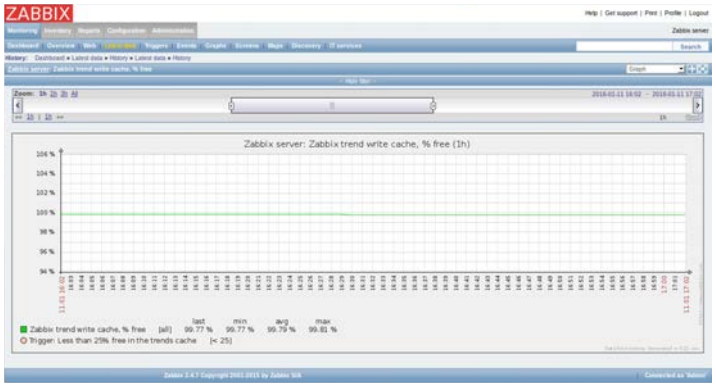


Figura 38. Caché de escritura en Zabbix Server (trends)

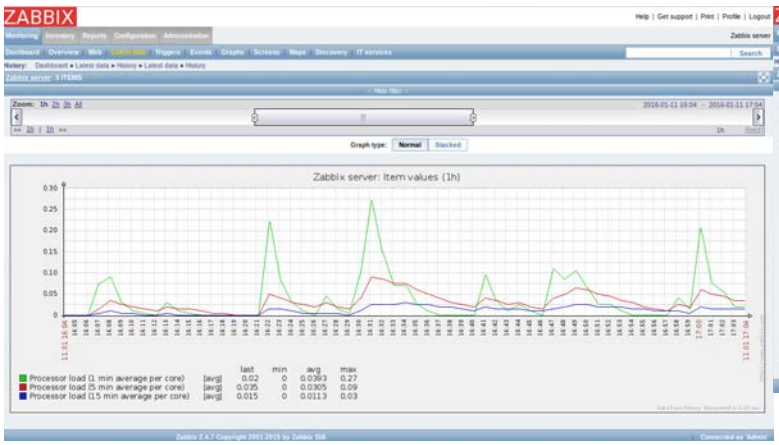


Figura 39. Gráfico conjunto de la carga de procesador en Zabbix Server

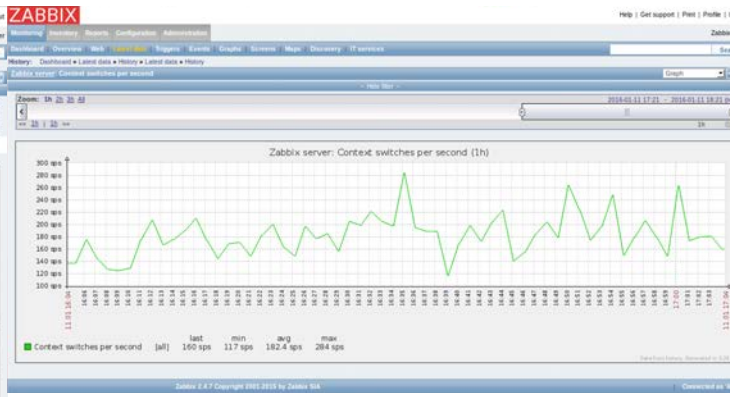


Figura 40. Cambios de contexto por segundo en Zabbix Server

Router UAV

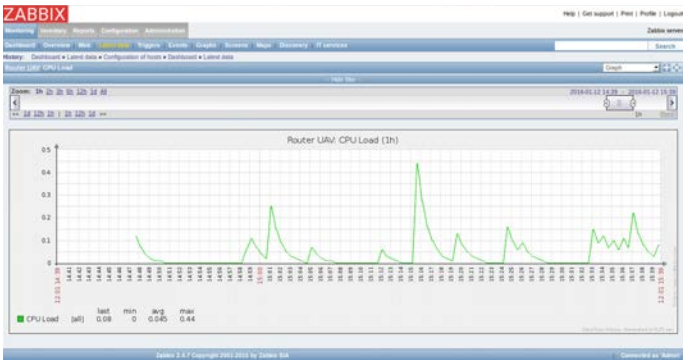


Figura 41. Carga de la CPU de Router UAV

Figura 42. Cambios de CPU de Router UAV

Figura 43. Utilidad de la CPU de Router UAV

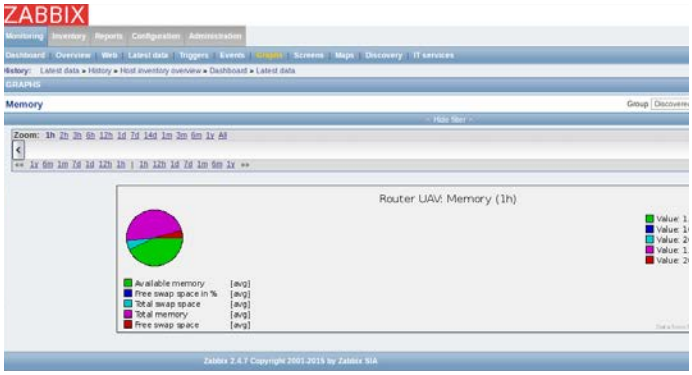


Figura 44. Gráfico personalizado tipo "pie" de la memoria de Router UAV

Como se puede observar, se han incluido diversos gráficos que ofrecen gran cantidad de datos e información sobre los parámetros a evaluar. Además, es posible personalizar y manejar los gráficos al antojo de cada uno, pudiendo hacer zoom en las zonas más llamativas, creando todo tipo de gráficos (barras, líneas, circulares …), así como elegir los intervalos de tiempo de cada gráfico – aspecto que se verá más adelante.

Zabbix es, por lo tanto, la mejor herramienta de monitorización de red para el despliegue y la que se elige por sus óptimas características. En apartados posteriores se hablará del diseño de la solución de nuestro problema con esta herramienta, su implementación y las pruebas y validaciones llevadas a cabo para poder llegar a obtener gráficos de datos obtenidos con un router UAV para concluir el proyecto.

3.3 Diseño e implementación

3.3.1 Introducción

A continuación se va a detallar el diseño del sistema de gestión de red objeto del presente TFG, así como su implementación, que estará basada en el uso de la herramienta de monitorización Zabbix. Será llevada a cabo en el despliegue de pruebas basado en le que se ha visto en el apartado anterior y tendrá como objeto mostrar como esta herramienta monitoriza todo tipo de parámetros y datos obtenidos de varios dispositivos relacionados con un vehículo aéreo no tripulado.

3.3.2 Diseño de la solución

En este apartado se va a explicar el diseño elegido para la solución, basándonos en la prueba de evalución preliminar sobre el sistema de gestión de red para vehículos aéreos no tripulados.

En primer lugar, se va a mostrar una figura de un potencial despliegue con vehículos aéreos no tripulados y su correspondiente sistema de gestión de red.

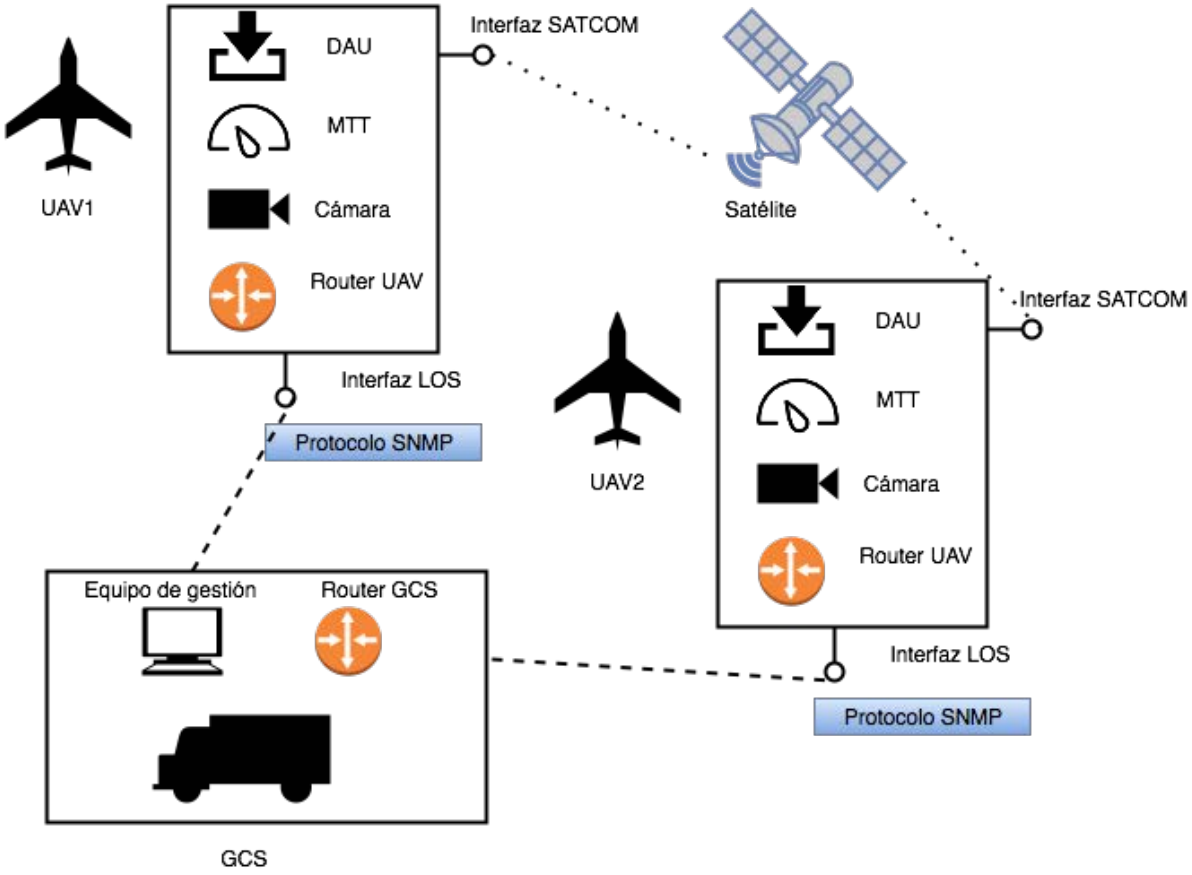


Figura 45. Diseño del despliegue

Como se puede apreciar, el diseño consta de varios UAVs (en la imagen sólo aparecen dos, pero pueden ser más) cuyos componentes principales son:

- Router UAV
- Unidad de adquisición de datos, también llamada DAU (*Data Adquisition Unit*)
- Unidad de control, llamada MTT
- Cámara
- Otros componentes

No todos los UAVs tendrían que ir equipados con los últimos componentes, siendo más importantes el router UAV y las interfaces de monitorización LOS (comunicación con la estación base) y SATCOM (para la comunicación con el satélite).

Con respecto a la estación base, o GCS (*Ground Control Station*), ésta consta de un router GCS y de un equipo de gestión que actuará como servidor que monitoriza los distintos dispositivos. Entre la estación base y los distintos UAVs se utilizará el protocolo SNMP para la gestión de los datos obtenidos, ya que como bien se ha visto, es el protocolo estrella para la gestión de red.

Con esto, se ha observado cómo se ha pensado que será el diseño de la solución en conceptos de despliegue. A continuación, se explicarán los requisitos que se necesitan para el posterior desarrollo del sistema de gestión de red:

- Capacidad de datos con un periodo configurable. Como se ha mencionado en el apartado del despliegue preliminar, es importante la necesidad en estos despliegues de obtener los datos en intervalos de tiempo pequeños. El tiempo de vuelo para los UAV puede ser variable, desde minutos hasta pocas horas, pero es importante que los datos que se recojan en periodos de tiempo cortos, para recoger la mayor cantidad de éstos y tener más información que representar luego en gráficas.

- Capacidad de generación de gráficas con los valores obtenidos, con actualización dinámica de estas. En este despliegue es fundamental el ver los cambios y la información obtenida de forma rápida y en tiempo real. No hay que olvidar que la gestión de la red de este despliegue tiene como objeto principal el de monitorizar la red y sus componentes y anticiparse antes los potenciales peligros para preverlos y evitarlos en medida de lo posible.

- Capacidad de flexibilidad para incorporar de manera sencilla la recogida de nuevos datos desde nuevos dispositivos (por ejemplo, nuevos equipos de comunicaciones en un UAV, nuevos UAVs, despliegues basados en máquinas virtuales, …).

- Basado en protocolos estandarizados, en particular el protocolo seleccionado para el intercambio de información de gestión SNMP.

- Basado en herramientas de software libre u Open Source disponibles en el mercado. Es fundamental para no tener que aplicar un coste al tener que comprar licencias que exorbiten el presupuesto del proyecto.

- Posibilidad de recoger datos no disponibles en la MIB de SNMP. Para este despliegue, hay que pensar en no sólo los datos disponibles para la red, sino también en aspectos relacionados con los UAV, por ejemplo, sensores de temperatura, humedad, configuraciones de los distintos componentes … es decir, estadísticas de datos que no utilicen el protocolo SNMP para el intercambio de información.

- Capacidad de mostrar los gráficos de forma conjunta y global, para su posterior estudio de manera rápida y sencilla.

- Capacidad de dar prestaciones de rendimiento óptimas teniendo en cuenta el enlace disponible y los posibles componentes que se incorporen en el UAV.

Por otro lado, los distintos parámetros a tener en cuenta en el despliegue se dividen en dos, los que pueden ser recogidos con el protocolo SNMP, y los que tienen que ser recogidos a través de otro modo de empleo, como son los scripts externos. Los parámetros que pueden ser recogidos a través de SNMP son:

- Porcentaje de CPU. En este parámetro se desea ver el uso de CPU al igual que se observa al usar el comando **top** de Linux.
- Porcentaje de Memoria. En este parámetro se desea ver la memoria total, utilizada y libre disponible para el dispositivo gestionado. También puede ser consultado con el comando **top** de Linux.

- *Throughput* LOS. Este parámetro medirá el *throughput* de entrada y salida para la interfaz LOS, que es la que comunica con la estación base.
- *Throughput* SATCOM. Este parámetro medirá el *throughput* de entrada y salida para la interfaz SATCOM, que es la que comunica con el satélite.

Y los recogidos a través de scripts son:

- Un set de parámetros que determinan el alcance de los distintos componentes del UAV a través de **ping**.
- AGC (*Automatic Gain Control*). Este parámetro se basa en el valor AGC que controla la ganancia de amplitud del voltaje de salida del dispositivo gestionado.
- RTT. Este parámetro permite ver el alcance a través de **ping** a la dirección IP interna del router UAV.
- Direcciones IP del router UAV. Este parámetro se usará para mostrar la tabla de direcciones IP del dispositivo gestionado.
- Tabla de rutas del router UAV. Este parámetro se usará para mostrar la tabla de rutas del dispositivo gestionado.

Con todo esto, ya se puede poner en práctica la implementación de dicho despliegue eligiendo la herramienta de gestión de red oportuna.

3.3.3 Desarrollo de la solución

Como se ha visto tras la evaluación del desarrollo preliminar, se elige Zabbix como herramienta de gestión para el sistema de gestión de red, porque cumple todos los requisitos del diseño, y por su lado, Cacti, tiene algunas limitaciones en estos requisitos como por ejemplo, la configuración del intervalo de recogida de datos, fundamental en el despliegue.

A la hora de decidir que desarrollo se le iba a dar a este proyecto, se pensaron en dos opciones principales. La primera consistía en instalar tanto agente como servidor Zabbix en el mismo dispositivo gestionado, en nuestro caso, en el router UAV. A priori, no es una práctica recomendada, pues al tener las dos entidades en el mismo dispositivo si se diese la hipótesis de que el dispositivo fallase o se estropease, se presentaría el problema de que ni servidor ni agente funcionarían. Sin embargo, la documentación de Zabbix recomendaba instalar servidor y agente a la vez, aunque después estuviese alguno de los dos inactivo. La ventaja sería que si el servidor cayese y estuviese en otra entidad, no habría comunicación y el escenario no sería válido. Esto es lo que se ha probado en el diseño.

Por otro lado, la segunda opción consistía en la instalación típica de agente en un dispositivo y servidor en otro. Esta opción sería la más lógica en un despliegue real, en el que el servidor se encontraría en una estación base y el agente en el mismo drone. Se decidió por lo tanto que la opción segunda era la mejor y más adecuada para el objetivo principal, y de esta manera, se llevó a cabo el despliegue de la herramienta en los distintos escenarios.

El primer despliegue se basa en un escenario básico donde un ordenador portátil ejerza de servidor y el router UAV sea el dispositivo agente a monitorizar. En este despliegue, se ha tenido que añadir un cable de red en la interfaz **eth2** del router UAV para obtener estadísticas de esta interfaz, además de la previa configuración que hicimos de **eth0**. Cabe destacar

también que en este despliegue se ha simulado el despliegue final con el router GCS y los demás dispositivos necesarios en el despliegue del UAV, haciendo que el router UAV funcione como UAV1, UAV2 y GCS. Se ha comprobado la conectividad entre agente y servidor con el comando ping y funciona a la perfección.

En definitiva, el servidor actuará como interfaz gráfica para el usuario y mostrará las gráficas creadas con la consola Zabbix y el agente enviará los datos de los parámetros no SNMP recogidos a través del enlace, haciendo que el servidor los procese y posteriormente los muestre en las gráficas.

Instalación de servidor Zabbix

En este apartado se ha de tener en cuenta la tabla con la instalación de Zabbix que se puede consultar en el Anexo III en el apartado de Zabbix. La instalación del servidor Zabbix es prácticamente análoga en este caso. Lo único que cambia es que al elegir el PC de pruebas como alojamiento del servidor, la distribución pasa de ser Debian a ser Ubuntu 10.02. Con esto, algunos de los pasos varían, teniendo en cuenta que hay que utilizar distintos paquetes.

Instalación de agente Zabbix

Este apartado es el que se llevo a cabo al principio, en la instalación de servidor y agente en el mismo dispositivo (router UAV) y es meramente seguir los pasos al pie de la letra de la tabla de instalación de herramientas (Anexo III, apartado Zabbix).

Lo importante en este paso es habilitar al servidor Zabbix del PC para que conecte con el agente Zabbix del router. Esto se logra modificando el fichero de configuración del agente Zabbix en el router. Este fichero se llama *zabbix_agentd.conf*, y se encuentra en el directorio `/etc/zabbix/zabbix_agentd.conf`. Se puede modificar fácilmente con el lector **vi** o **nano**. En este caso se ha utilizado **vi** y se han modificado los siguientes parámetros:

```
Server: [dirección IP del servidor, en nuestro caso 10.1.2.2]
ServerActive: [dirección IP del servidor, 10.1.2.2]
Hostname: ZS_terminal [nombre para el dispositivo]
Port: 10050 [puerto por defecto]
```

Una vez insertados, se guarda y se reinicia el agente de la siguiente forma:

```
root@drone-idan-uav:~$ /etc/init.d/zabbix-agent restart
```

Con esto, Zabbix ya es capaz de establecer conexión entre agente y servidor que se encuentran alojados en dispositivos diferentes. En el caso del despliegue, el agente Zabbix no será utilizado nada más para ejecutar dos tipos de scripts creados, pues se ha elegido así. No está de más tenerlo instalado por si en algún futuro nos gustara recoger otro tipo de estadísticas que dependan del agente.

Creación de *host*

Para crear un *host* en este escenario, se seguirá la tabla de utilización de Zabbix (Tabla 4). De la misma forma, se dará un nombre al *host* creado y se le asignarán las interfaces por las que escuchará. En esta creación hemos de crear dos tipos de interfaz: la del agente y la de SNMP, pues queremos obtener valores a través de parámetros SNMP como la CPU, la memoria, el tráfico por interfaz, …

En el campo **Agent Interface**, se pone la IP del router UAV (10.1.2.1) y el puerto por defecto (10050) y el el campo **SNMP Interface**, de manera análoga, la IP del router UAV (10.1.2.1) y el puerto SNMP por defecto (161). Una vez introducidos estos datos, se crea el host y se habilita en la pestaña **Enable/Disable**.

Como se observó en apartados anteriores, ahora es momento de crear los *Items* que servirán de parámetros para el *host*.

Adquisición de parámetros no SNMP (Scripts externos)

Para la adquisición de parámetros no SNMP hay que basarse en el paso de creación de ítems de la tabla 4. La creación de parámetros no SNMP servirá para incluir los valores obtenidos por otros medios, como por ejemplo, el valor AGC, obtenido a partir de un script externo u otros valores procedentes de scripts – como la tabla de rutas IP o la de direcciones IP – que serán scripts que se ejecuten en el agente.

La utilización de scripts en Zabbix para obtener datos de valores externos es muy amplia y es por ello que se ofrecen herramientas dentro de ella para poder obtener todo tipo de datos.

En este caso, en primer lugar se crea un pequeño script que dé valores aleatorios en un intervalo de cinco segundos, simulando un pequeño programa en el que el valor AGC se vuelca en un fichero y después necesita ser leído. Este script se incorporará a la consola Zabbix por el mecanismo llamado **External Check**. Este mecanimo sirve para chequear scripts que recojan estadísticas sin tener ningún agente instalado en el dispositivo. El script se guarda en la carpeta `/usr/local/share/zabbix/externalscripts/agc` (*agc* es el nombre que se le da al script).

Una vez hecho esto, se vuelve a reiniciar el agente y ya se puede incluir como parámetro a representar. Para ello, vamos a **Configuration – Hosts – Host deseado – Items – Create ítem**. Dentro de esa pestaña, creamos un ítem igual que en anteriores ocasiones eligiendo tipo **External Check**. Pulsamos **Add** y listo.

Además de este script que devuelva valores AGC simulando el programa que lo monitoriza, se crearán otros para los pings y la pérdida de paquetes para RTT (*Round Trip Time*), MTT, Camera y ACRA, dispositivos del despliegue. Para ver todos los scripts en detalle, se puede consultar la tabla de scripts del Anexo IV.

Por otro lado, se crearán scripts que vayan sobre el agente Zabbix que está en el router UAV. Estos scripts serán los que recojan la tabla de rutas IP y la de direcciones IP. Para estos scripts, es necesario modificar el fichero de configuración del agente Zabbix, en `/usr/local/etc/zabbix_agentd.conf.` Lo que hay que modificar en este fichero es lo que Zabbix llama **User Parameters**, que consisten en chequeos del agente Zabbix de parámetros que no vienen por defecto con la instalación de la herramienta.

En el caso del despligue, se crearán dos scripts, uno para recoger la tabla de direcciones IP y otro para recoger la tabla de rutas. Estos scripts se guardarán en `/usr/local/etc/scripts/[nombre del script]` –se eligen **ifconfig** y **ipro**, respectivamente. Cuando estén guardados en dicho directorio, hay que conceder todos los permisos correspondientes. Eso se hace con el comando **chmod**, otorgando todos los permisos a todos los scripts guardados en dicho directorio. El siguiente paso será modificar los **User Parameters** en el archivo de configuración del agente Zabbix. Con el comando **vi**, se modifica el archivo incluyendo las siguientes líneas en el apartado **User Parameters**:

```
UserParameter = ifconfig, /usr/local/etc/scripts/ifconfig
UserParameter = ipro, /usr/local/etc/scripts/ipro
```

Se guarda y se reinician agente y servidor Zabbix. Una vez hecho esto, para empezar a recoger datos de los scripts creados, se crean los ítems de la misma forma vista con anterioridad, teniendo en cuenta que en el campo **Key** se pondrá el nombre que le dimos al User Parameter (**ifconfig** o **ipro**). Para más información sobre estos scripts, se puede consultar de nuevo el Anexo IV.

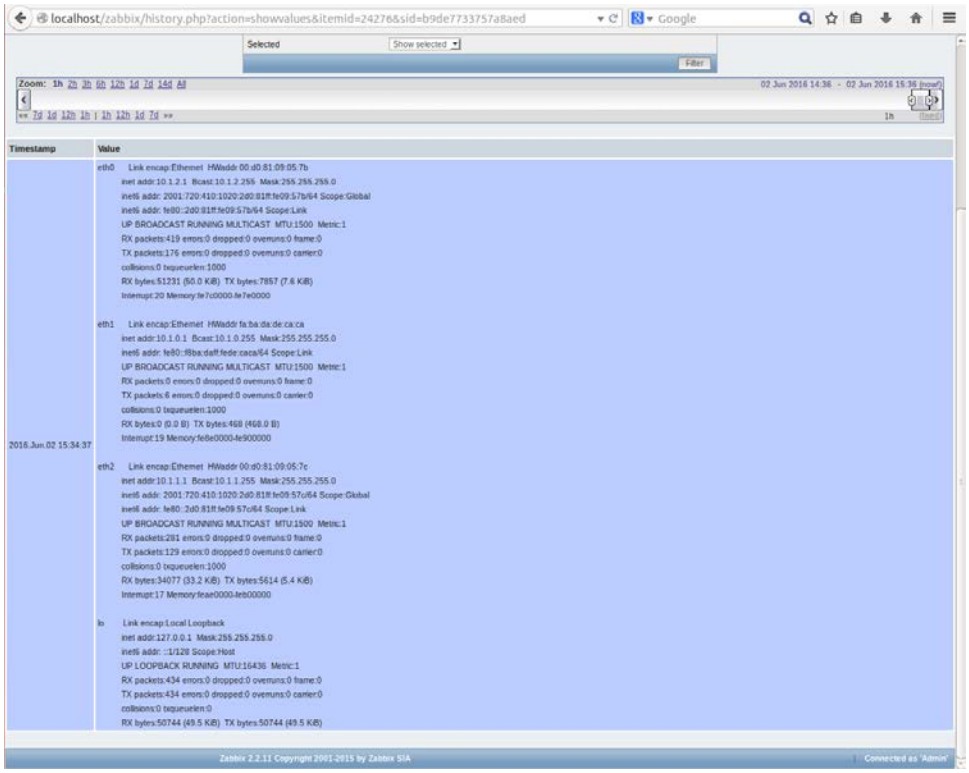


Figura 46. Item para script ifconfig

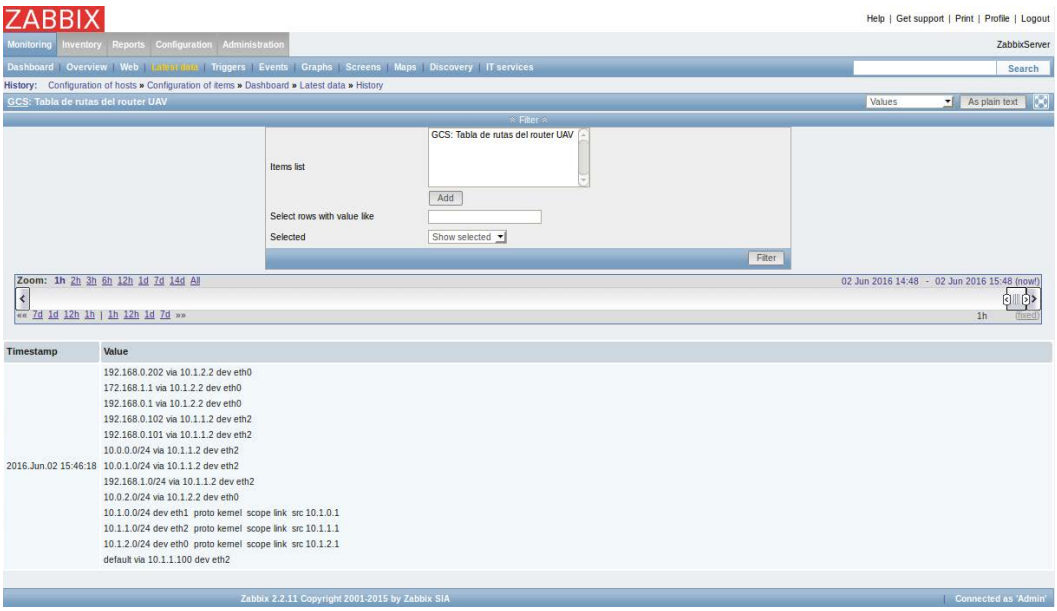


Figura 47. Item para script ipro

Otra forma de crear parámetros no SNMP sería importar los que vienen por defecto en Zabbix (los *template*). Para eso sí que es necesario un agente Zabbix. Estos parámetros en realidad son scripts previamente instalados con la herramienta que miden multitud de parámetros y ofrecen los resultados en gráficas. Sin embargo, el despliegue actual se centrará en los creados a partir de scripts importados previamente creados y en los parámetros SNMP.

Creación de parámetros SNMP

La creación de parámetros SNMP es ligeramente distinta a la creación de parámetros no SNMP. La principal diferencia aquí es que hay que tener en cuenta que versión de SNMP soporta el dispositivo gestionado y de esa manera, obtener los OID correspondientes a cada parámetro que queremos representar. En principio, no es tarea difícil, pero si más tediosa que la de crear parámetros no SNMP, pues estos se pueden crear a partir de plantillas ya configuradas por Zabbix.

Teniendo en cuenta que el dispositivo en el despliegue de pruebas corre sobre una distribución Linux, hay que conocer que OID's son los soportados por este sistema operativo [Linux OID, 2006]. Una vez obtenida una lista con los parámetros que se desean monitorizar, se comprueba con la herramienta **snmpget** – propia de SNMP – los valores que se están obteniendo, de la forma:

```
snmpget –v [versión de SNMP] –c [nombre de comunidad] [dirección IP del dispositivo] [OID del parámetro]
```

En este caso, la versión es 1, el nombre de comunidad es "public", la dirección IP es 10.1.2.1 y el OID el que obtengamos de la lista previamente elaborada. Cuando esto sea satisfactorio, se pueden crear los parámetros SNMP a monitorizar en Zabbix. Al igual que en anteriores ocasiones, se ha de crear el *host*, esta vez asegurándonos que escuche por la interfaz SNMP y hecho esto, crear los ítems oportunos que serán los parámetros a graficar.

A la hora de crear el ítem, hay que tener en cuenta tres campos importantes: el tipo de agente (será SNMP Agent versión 1 en este caso), el interfaz y puerto (10.1.2.1:161) y el OID (que se obtendrá de la lista contemplada en la sección de Anexos). Con esto, se dejan marcados los campos que vienen por defecto y se crea el ítem. Una vez hecho esto, sólo hay que esperar a que se recojan datos y se empiecen a mostrar en las gráficas. La lista de OID's implementados para este desarrollo se puede consultar en la sección de Anexos (Anexo V).

Screen final

En primer lugar, el escenario que se va a tener en cuenta a la hora de diseñar la pantalla de parámetros consiste en tres equipos de red distintos monitorizados por Zabbix:

- El router de la estación base, o GCS
- Un router llamado UAV1 para primer UAV
- Un router llamado UAV2 para un segundo UAV

En la pantalla de monitorización principal habrá tres hosts distintos cuyos parámetros serán los siguientes. Para el router GCS:

- % CPU
- % Memoria
- Throughput LOS
- Throughput SATCOM
- Direcciones IP de la GCS
- Tabla de rutas de la GCS

Para los UAVx (siendo x = 1, 2):

- Un primer set de 4 parámetros que determinan el alcance (0 ó 1 en función de que funcione o no el ping). Se llamarán: routerUAV, MTT, Camera, ACRA.
- AGC
- % CPU
- % Memoria
- RTT (a la dirección IP interna del router UAV)
- Throughput LOS
- Throughput SATCOM
- Direcciones IP del router UAV
- Tabla de rutas del router UAV

Todos los parámetros de los equipos de red se monitorizarán en un intervalo de 5 segundos, excepto los *throughput*, que se medirán cada 30 y las tablas de direcciones y rutas, que se monitorizarán cada 10 minutos. La elección de estos intervalos ha sido realizada tras una batería de pruebas para calcular con que *threshold* se ofrecen datos verídicos y fiables sin perder garantías, a la vez de obtener los máximos datos posibles. Estas pruebas se verán en apartados posteriores. Para más información sobre los OID concretos utilizados para la recogida de estos parámetros se puede consultar de nuevo la lista en el apartado de Anexos (Anexo V).

Además, se contará con distintos dispositivos conectados por la red, como la cámara, ACRA y MTT. Se crearán dos tipos de *screens* distintas, una para los router UAVx y otra para el router GCS.

Router GCS		**Router UAVx**	
Throughput LOS	Throughput SATCOM	Throughput LOS	Throughput SATCOM
AGC	RTT	% CPU	% Memory
% CPU	% Memory		

Tabla 5. Organización de las *screens*

En algunas de las gráficas de las *screens*, como en las mediciones del throughput, RTT y porcentaje de memoria, se han tenido que crear gráficas compuestas. Éstas se crean en **Configuration – Hosts**, eligiendo el host deseado donde crearlas y después pulsando **Graphs – Create New Graph**.

Tomando como referencia los parámetros configurados previamente para cada host, se crean tres gráficos distintos, útiles para cada *screen*:

- Un gráfico compuesto para el *Throughput* LOS, con el throughput de entrada y de salida para la interfaz LOS. Pinta en azul el tráfico de entrada y en amarillo el de salida.
- Un gráfico compuesto para el *Throughput* LOS, con el throughput de entrada y de salida para la interfaz SATCOM. Pinta en azul el tráfico de entrada y en amarillo el de salida.
- Un gráfico compuesto para el RTT, con el RTT y el RTT Loss, parámetros que se recogen a través de scripts previamente incluidos. Pinta en verde el RTT y en rojo la pérdida de paquetes.
- Un gráfico compuesto para el porcentaje de memoria, con la memoria usada, la memoria libre y la memoria total. Pinta en azul la total, en verde la libre y en rojo la usada.

Una vez hecho esto – se pueden crear los gráficos para uno de los *hosts* y después copiarlos en el otro con el botón **Copy Selected To** – ya se pueden crear las *screens*. En el mismo *host*, hay una pestaña llamada ***Screens***. Ahí, pulsamos en **Create *Screen*** y de forma similar a la creación de tablas en cualquier programa de ofimática, elegimos las filas y columnas que se deseen para la *screen*. Una vez elegido esto, se añaden las gráficas según el orden deseado. Si son gráficas simples se elige **Simple Graphs**, mientras que si son de las gráficas compuestas creadas a partir de varios parámetros (como la de RTT, *throughput* o memoria), simplemente se elige **Graphs**. Cuando se añaden se puede ver todas las gráficas de forma general como en las siguientes imágenes. Pinchando en cada gráfico, estos se amplían para poder ver la gráfica de manera más detallada.

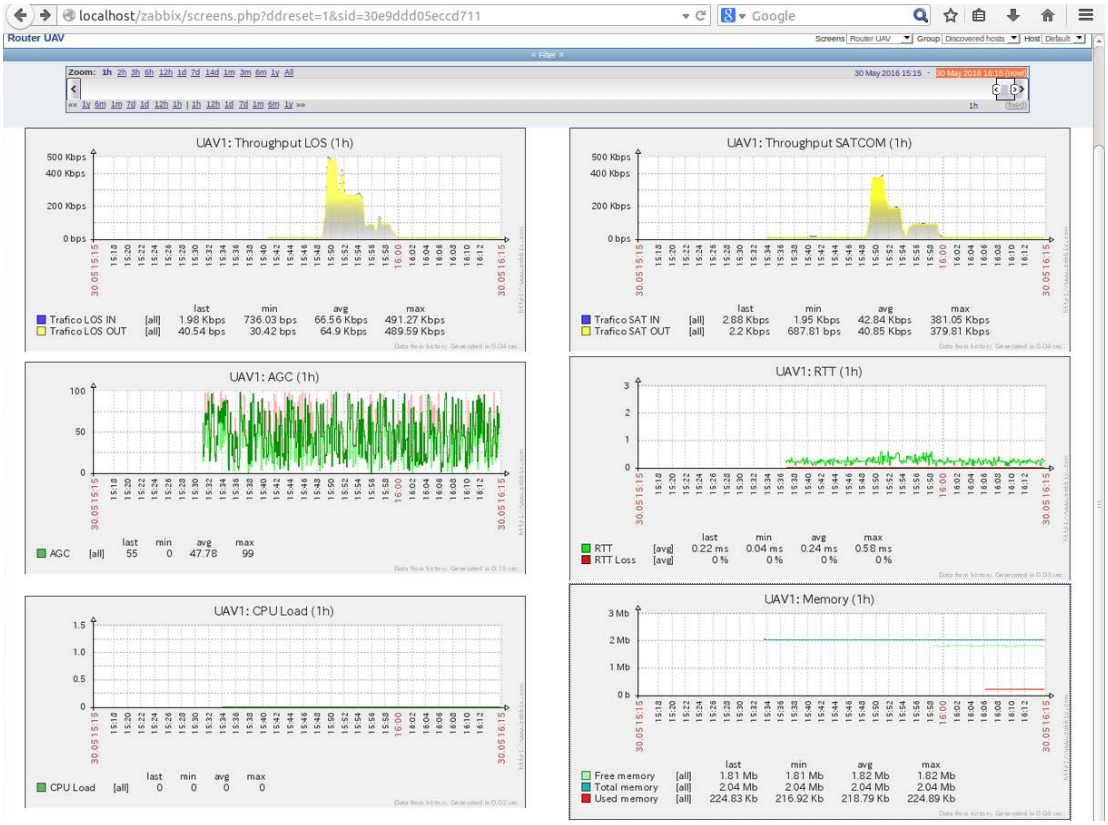


Figura 48. *Screen* router UAV

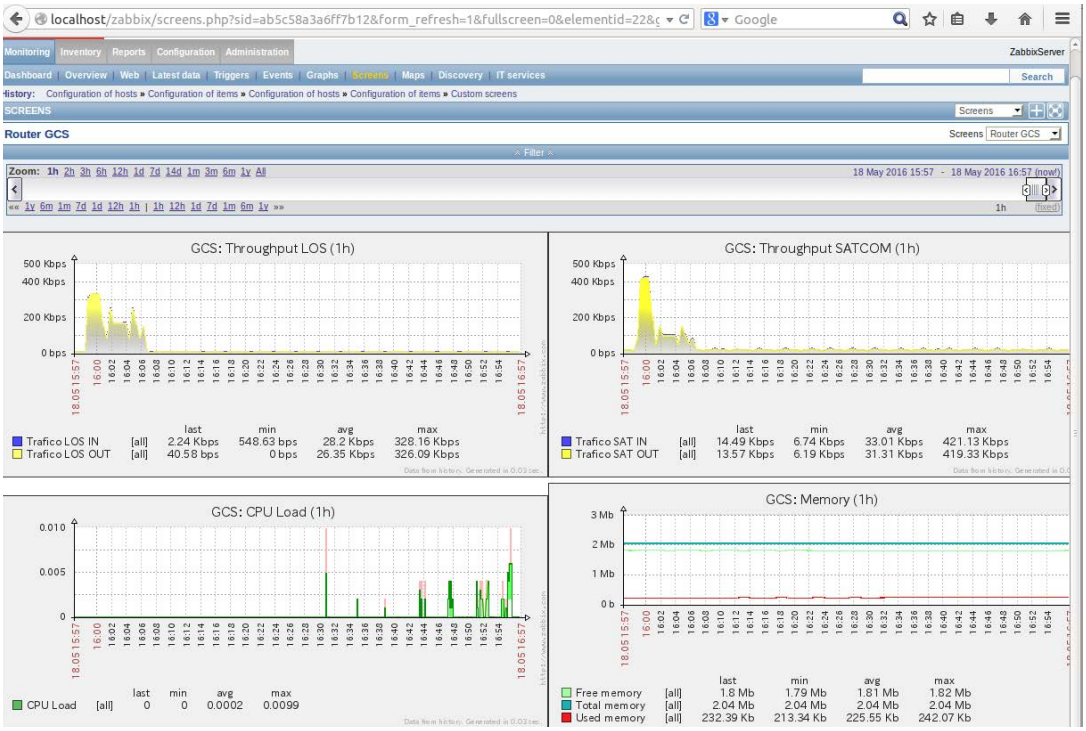


Figura 49. *Screen* router GCS

Este método es bastante útil si se desea monitorizar varios elementos a la vez y tener constancia de todos ellos de manera rápida y visual. Para ver las gráficas más detalladas, basta con pinchar encima de ellas y se observará la vista detallada en la que se pueden ver las gráficas mejor.

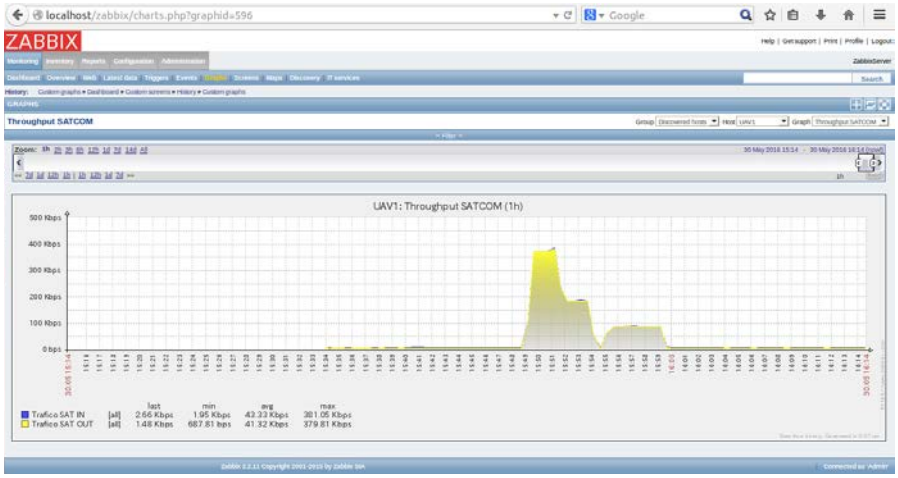


Figura 50. Vista en zoom de gráfica de la *screen* (*Throughput* SATCOM)

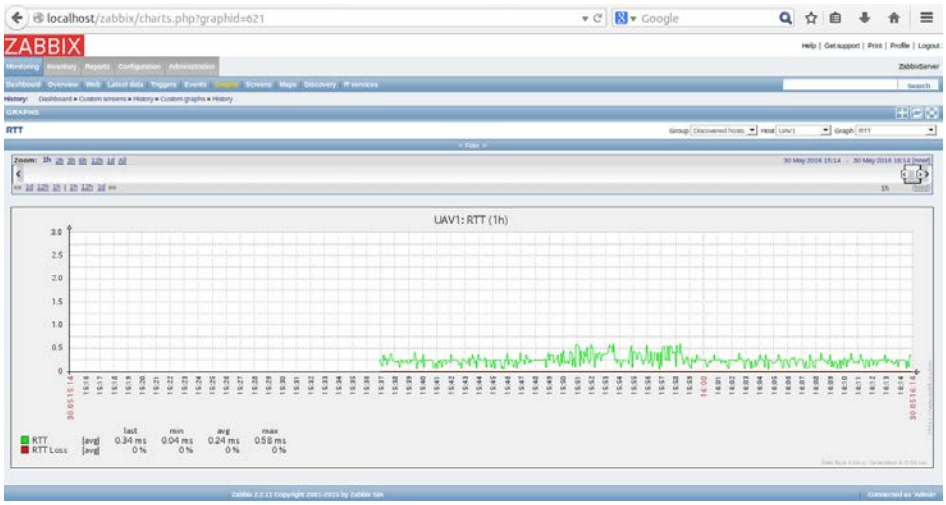


Figura 51. Vista en zoom de gráfica de *screen* (RTT)

Capítulo 4

Pruebas, resultados y evaluación

4.1 Introducción

En este apartado se van a realizar una serie de pruebas para poner a punto la herramienta de gestión de red elegida, Zabbix, y ver cómo se comporta en determinadas situaciones, para medir el rendimiento y la optimización que se puede conseguir a la hora de usarla como herramienta de gestión de redes.

En primer lugar, se hará una pequeña prueba inicial para ver que el tráfico que se captura por las interfaces es en realidad tráfico SNMP, y ver si es correcto su comportamiento, actuando dentro de los intervalos de tiempo establecido y mandando los datos correctos.

La segunda batería de pruebas consistirá en cuatro pruebas unitarias para medir el uso de la CPU y el consumo de ancho de banda. Cada prueba durará aproximadamente tres minutos, siendo los requisitos de cada una los siguientes:

1. Prueba consistente en la monitorización de 10 parámetros mediante SNMP, con un periodo de recogida de datos de 1 segundo.
2. Prueba consistente en la monitorización de 40 parámetros mediante SNMP, con un periodo de recogida de datos de 1 segundo.
3. Prueba consistente en la monitorización de 10 parámetros mediante SNMP, con un periodo de recogida de datos de 10 segundos.
4. Prueba consistente en la monitorización de 40 parámetros mediante SNMP, con un periodo de recogida de datos de 10 segundos.

Estos requisitos – parámetros e intervalos – servirán para calcular el *threshold* o umbral de optimización, es decir, el número de parámetros y longitud de intervalos que dan los valores más óptimos teniendo en cuenta las características del despliegue, observando y estudiando cómo afectará al router UAV que Zabbix escoja más o menos parámetros para representar en intervalos variables, y de esta forma ver si afecta mucho al consumo tanto de CPU como de ancho de banda.

Es importante recalcar, que el enlace que posee el router UAV no sólo se utilizará para la monitorización de la red, sino también para comandar el vehículo aéreo y para la transmisión de vídeo, así como otros parámetros de telemetría, por lo tanto, es importante medir

parámetros como el consumo de la CPU y del ancho de banda para ver cómo afectará de forma global al enlace y al despliegue.

Finalmente, se realizarán distintas pruebas para valorar otras plataformas para desplegar los ditintos componentes del sistema de gestión, en particular máquinas virtuales o plataformas hardware de tamaño reducido (Raspberry Pi). Esto servirá para comprobar que Zabbix es realmente la herramienta versátil y flexible útil para este proyecto.

4.2 Pruebas

4.2.1 Pruebas de parámetros SNMP

Como se ha visto en los apartados de Diseño y Desarrollo, la mayoría de los parámetros utilizados para el despliegue son SNMP. Es por ello importante hacer la comprobación de que estos parámetros están siendo enviados y recibidos correctamente a través del enlace, y que dicha transmisión se hace dentro de los intervalos elegidos en Zabbix.

En una primera instancia, para comprobar que los parámetros enviados y recibidos a través del enlace son los correctos, se usa la herramienta **snmpwalk**. Esta herramienta permite hacer una consulta SNMP sobre el OID que se desee obtener, mostrando su valor. Por ejemplo, si se desea conocer las estadísticas de carga de CPU durante un minuto en la interfaz **eth0** del router UAV, se introducirá el siguiente comando:

```
snmpwalk –v 1 –c public 10.1.2.1 .1.3.6.1.4.1.2021.10.1.3.1
```

En la opción –v se escribe la versión de SNMP instalada (la 1), en –c el nombre de comunidad (*public*), seguido de la dirección IP de la interfaz sobre la que queramos consultar y finalmente, el OID a consultar.

Aparte de comprobar los valores obtenidos con **snmpwalk**, se han comprobado todos los OID concretos de los parámetros SNMP necesarios para el diseño y se ha verificado que todos dan los valores correspondientes. En el caso de la CPU y el uso de memoria se ha comprobado con el comando **top**, las pruebas de retardo con su comando característico (la herramienta **ping**), las tablas de IP’s y rutas con los comandos **ifconfig** y **ip ro** de Linux, respectivamente, y finalmente, para el *throughput*, se han hecho pruebas en las que se realizaban pruebas de retardo con la herramienta **ping** con tamaños de paquetes distintos y se mostraba en las gráficas de Zabbix correctamente.

Con esto, ya estaría completa la parte de comprobación de parámetros SNMP. Ahora se comprobará si los intervalos y lo qué realmente se envía y se recibe son parámetros SNMP. Para ello, se usará **Wireshark**. Las posteriores figuras muestran una captura de los mensajes correspondientes a la primera prueba unitaria de rendimiento que se realizará en la siguiente sección (se recogen 10 parámetros cada 1 segundo). Se observa la herramienta Wireshark interpreta dichos mensajes como son un *get-request* y un *get-response* de SNMP.

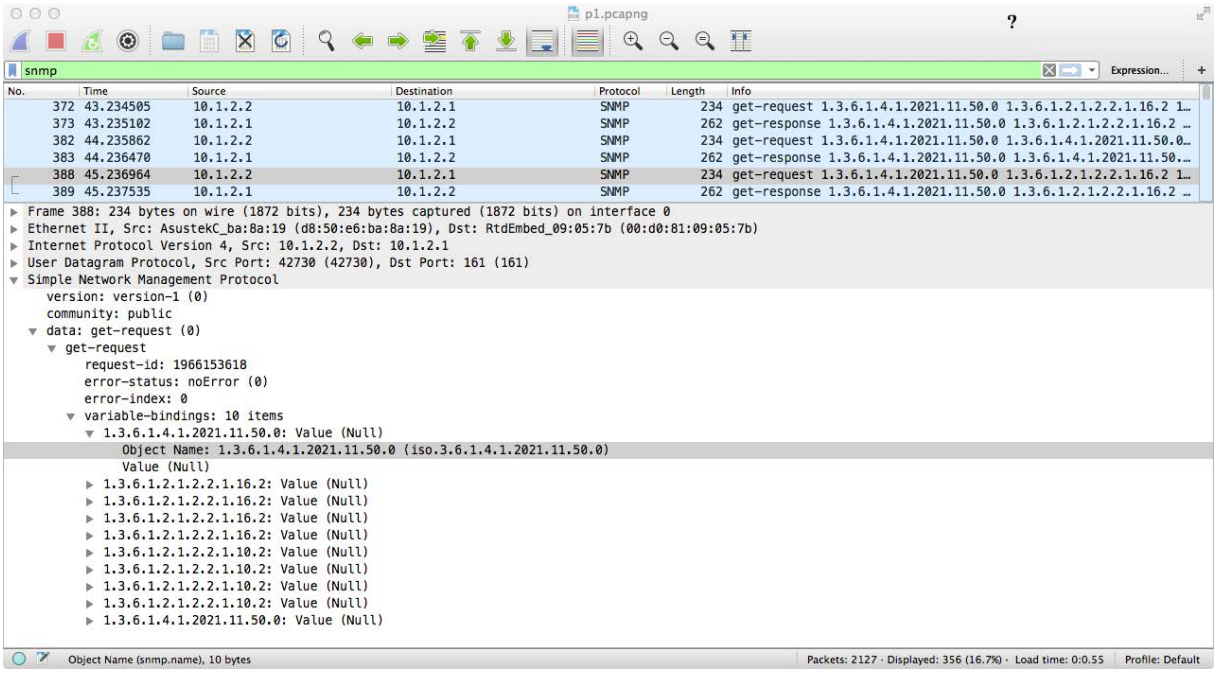


Figura 52. *Get-request* SNMP

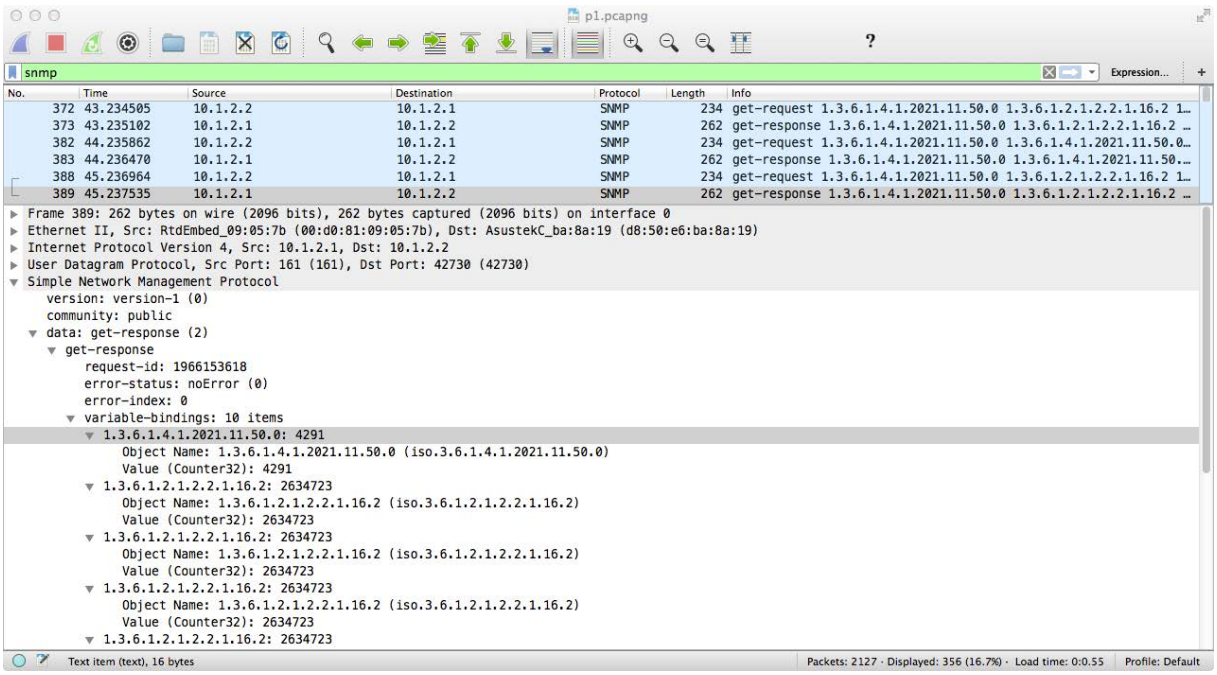


Figura 53. *Get-response* SNMP

Como se puede apreciar en ambas figuras, filtrando los paquetes por SNMP, se obtienen todos los *get-request* y *get-response* de esa sesión. Teniendo en cuenta la figura 71, se elige un *get-request* al azar y dentro de él, un OID también al azar (en este caso, es el que mide el *raw user cpu time* en la interfaz **eth0**). Se observa que en un principio está a *NULL* (no tiene valores) pues está dentro de un *get-request*, que lo que hace es consultar qué valor tiene. Como se ve posteriormente en la figura 73, se da su valor, 4291, al igual que al resto de OID's por los que se preguntaba. También se observa que el sentido de los mensajes es el correcto, ya que el *get-request* va del servidor al router UAV, y el *get-response* al revés. Es importante que entre

cada dupla *get-request/get-response* también se cumple el intervalo previamente establecido de un segundo.

El intervalo elegido para recoger datos SNMP es un concepto a tener en cuenta a la hora de trabajar con Zabbix ya que cabe recalcar uno de los pequeños problemas a los que se hizo frente a la hora de configurar el Diseño en el apartado anterior y que tiene que ver con el intervalo de tiempo. A la hora de elegir el intervalo para el *throughput* se observó que valores muy pequeños (de 1 a 20 segundos) hacían que las gráficas Zabbix no se comportasen correctamente, ofreciendo picos variables que cambiaban a medida que avanzaba el tiempo. Tras algunas pruebas en las que se probó a configurar intervalos desde un segundo a treinta, se comprobó que al configurar este parámetro en un intervalo mayor (30 segundos) se pudo ver que las gráficas se normalizaban y ofrecían valores correctos.

Esta prueba se puede considerar por tanto exitosa, y permite avanzar con el resto de comprobaciones.

4.2.2 Pruebas de rendimiento

4.2.2.1 Configuración previa para las pruebas

A la hora de ejecutar las pruebas de rendimiento, se han seguido una serie de pasos visibles en la siguiente tabla:

Número de paso	Descripción del paso a seguir
1	Cuando se inicia la consola de Wireshark, hay que empezar a escuchar en **eth0**, que es la interfaz conectada a la subred a la que pertenece el router UAV. Para filtrar los paquetes SNMP, hay que incluir la siguiente línea en el filtro: `snmp`.
2	Una vez Wireshark está funcionando, se pone en marcha la herramienta que servirá para hacer mediciones sobre el consumo de CPU. Esta herramienta será el comando **top** de Linux [Pérez Esteso, 2014]. Este comando ofrece muchísima información pero en este caso, el objetivo será basarse principalmente en los datos concernientes al consumo de CPU. En este aspecto, **top** muestra los porcentajes de uso de procesador diferenciado por usos (usuarios, sistema, procesos inactivos …). Este comando se ejecutará dentro del router UAV, ya sea que se conecte por él vía SSH o cuando se haga de forma directa a través de la pantalla conectada. El comando a introducir es el siguiente: `root@drone-idan-uav:~$ top` Cuando se escriba esto se ejecutará y empezará a mostrar la información correspondiente.
3	Hasta aquí, las configuraciones concernientes a herramientas externas a Zabbix. La configuración de Zabbix es bastante sencilla. Se pondrá como ejemplo el primer escenario de pruebas (10 parámetros en 1 segundo). Para configurarlo, se accede a los Items a través de **Configuration – Hosts – Items**. Ahí se nos despliegan todos

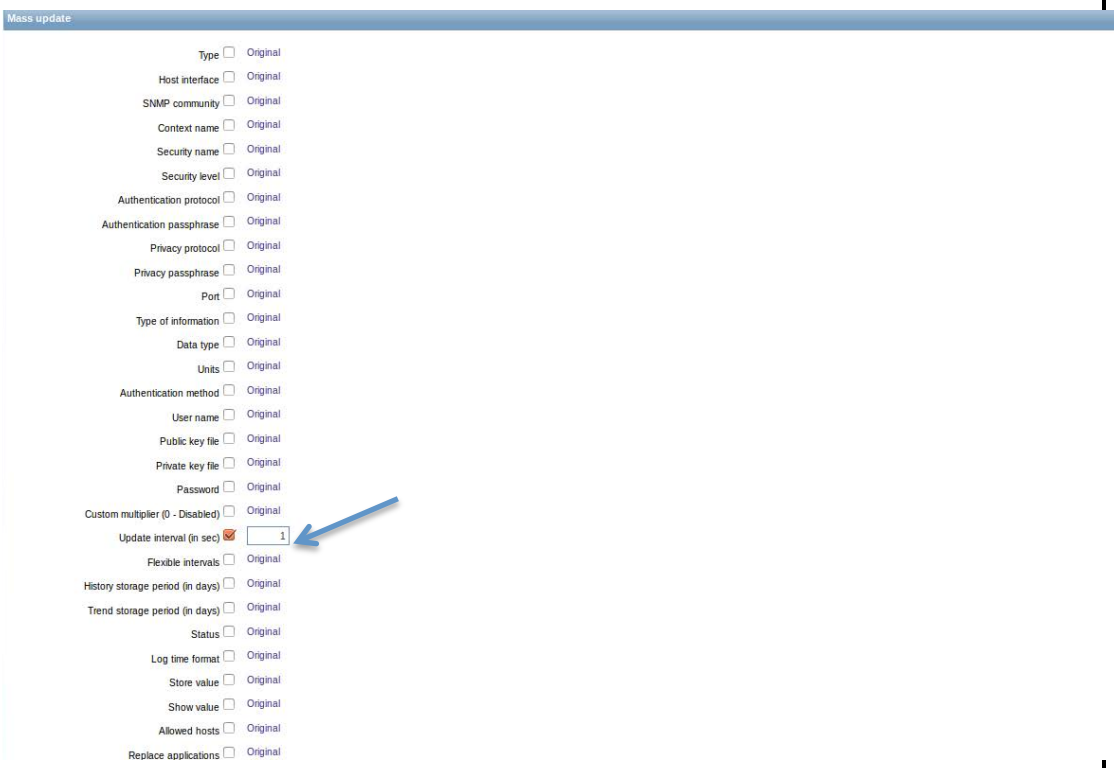
	los ítems correspondientes al host elegido (en nuestro caso, Zabbix server). Para este caso, se eligen 10 parámetros SNMP al azar y se dejan solamente éstos *Enabled* (para que sean los únicos de los que se representen gráficas). Para modificar el intervalo de actualización/recogida de datos se procede a hacer una **Mass Update**, o actualización masiva. Esto se hace marcando los 10 parámetros elegidos y después en el desplegable de opciones de la barra inferior elegir **Mass Update** y dar a **Go**.  **Figura 54.** ***Mass Update*** Como se puede apreciar, ahí se pueden modificar una gran cantidad de parámetros de forma masiva. Se elige el que interesa, *Update interval (in sec)*, para modificar el intervalo de representación de los datos en las gráficas Zabbix. Se elige un segundo y aplicamos los cambios.
4	Una vez hecho esto, si se espera un corto intervalo de tiempo y se va a la pantalla para ver las gráficas de Zabbix, se ve el cambio significativo. Los gráficos salen con más detalle y con líneas más irregulares, dando así la conciencia de que se representan los cambios de los parámetros cada poco tiempo. Con estos pasos ya está todo configurado para poder hacer las mediciones oportunas de CPU y ancho de banda en el enlace del router UAV.

Tabla 6. Configuración previa para las pruebas

4.2.2.2 Resultados

Primera prueba

La primera prueba consiste en monitorizar 10 parámetros al azar en intervalos de 1 segundo. Estos son los resultados:

Comencemos por el ancho de banda. En el caso que ocupa, hay que esperar a acabar la prueba para crear un gráfico que muestre cómo se ha comportado el consumo de ancho de banda, viendo un pequeño resumen de las estadísticas obtenidas en esa captura. Para ello, una vez finalizada la prueba, se pincha en **Statistics – Summary** (o **Statistics – Capture File Properties**, según la versión de Wireshark).

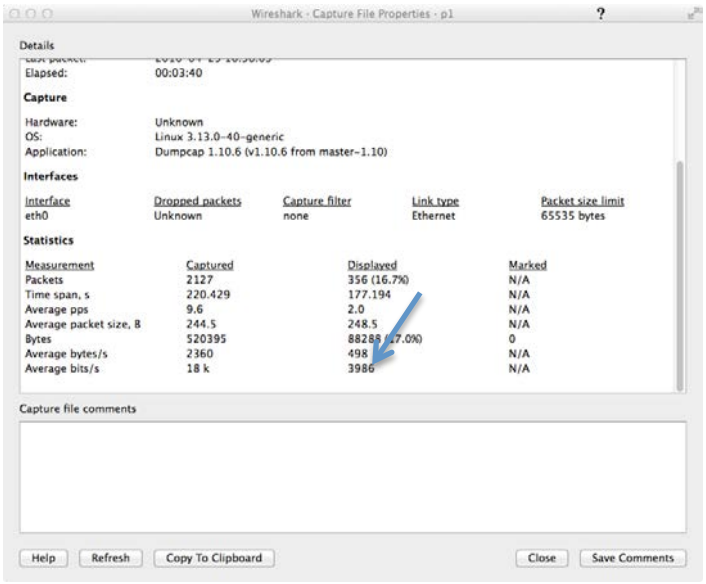


Figura 55. Medición de ancho de banda de la Prueba 1

Como se puede ver, hay multitud de estadísticas, pero la que nos interesa es la que está bajo la columna **Displayed** de la fila **Averages bits/s**. En esta se muestra la media de ancho de banda respecto a los paquetes filtrados. Evidentemente, 3986 bps es una media bastante prometedora pues equivale a 0,003986 Mbps, un consumo muy bajo con respecto al que provee el enlace. Sin embargo, hay que tener en cuenta todas las pruebas para estimar lo óptimo que parece este resultado.

En relación al consumo de la CPU, se obtienen valores fluctuantes pero constantes causados lógicamente por la entrada de bloques de datos a la vez, sobre todo cuando se mandan datos a la interfaz web de Zabbix o se hace uso de la base de datos. Ahí es cuando se nota más el consumo. En ocasiones el porcentaje de uso de la CPU en segundos es 0 o cercano a 0. Esto coincide en los momentos en que no se están mandando datos a Zabbix para representar.

Como se puede observar en la Tabla 10, se obtienen valores (medidos en %) muy razonables y óptimos, no críticos a la hora de hablar del consumo de CPU y claramente prometedores a falta del resto de las pruebas.

1			
user	sys	idle	wa

Media	0,220	0,152	99,540	0,105
Máximo	4,8	1,8	100,0	0,5
Mínimo	0,0	0,0	93,2	0,0

Tabla 7. Datos de consumo de CPU en Prueba 1

Segunda prueba

La segunda prueba consiste en monitorizar 40 parámetros al azar en intervalos de 1 segundo. Como se puede suponer a priori, éste sería el caso peor, pues es el caso que más parámetros evalúa en el intervalo de tiempo más pequeño, y que en principio, tendría más consumo de ancho de banda y de CPU. Al igual que la vez anterior, se mide el ancho de banda con Wireshark, con los mismos filtros y parámetros de configuración.

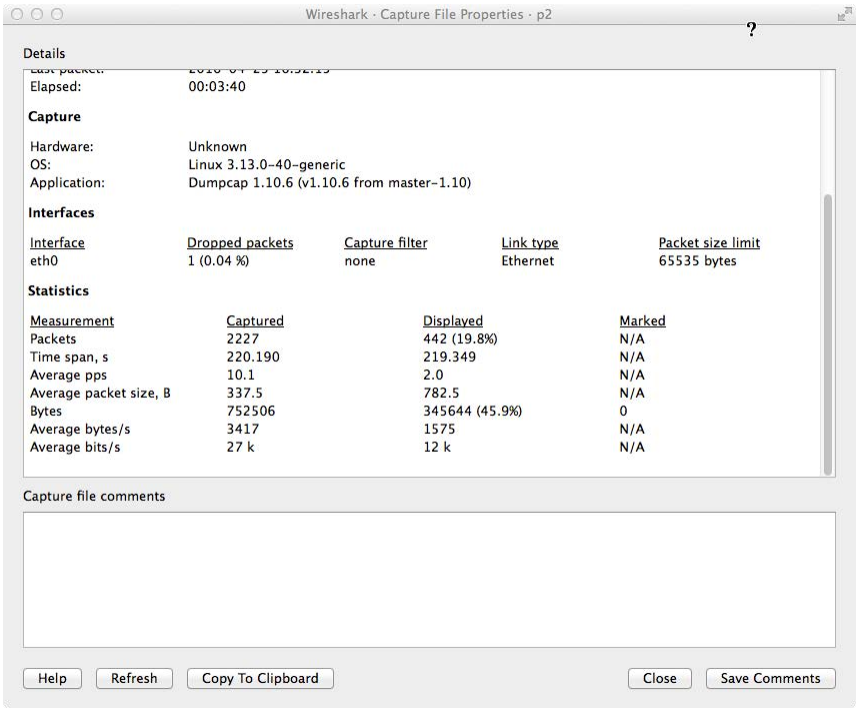


Figura 56. Medición del ancho de banda de la Prueba 2

Como se puede observar, se obtiene una media bastante más alta que en la primera prueba (0,012 Mbps). Este es el comportamiento esperado, ya que al incluir más parámetros en un intervalo de tiempo menor, el tráfico que se intercambia por el enlace es más constante y elevado, y por lo tanto, se ve reflejado en la media. Aún así, sigue siendo un consumo óptimo para el enlace existente.

En cuanto al consumo de CPU, también como en la primera prueba se observan fluctuaciones pero consumo constante. Lógicamente, al ser ésta el caso peor se ha de esperar obtener valores de consumo de CPU más altos, y efectivamente, es así.

2

	user	sys	idle	wa
Media	0,225	0,145	99,531	0,115
Máximo	6,5	1,8	100,0	0,5
Mínimo	0,0	0,0	91,4	0,0

Tabla 8. Datos de consumo de CPU en Prueba 2

Sin embargo, como se puede observar, siguen siendo valores aptos para este despliegue y además, muy poco variables con respecto a las pruebas ya realizadas.

Tercera prueba

La tercera prueba consiste en monitorizar 10 parámetros al azar en intervalos de 10 segundos. A priori, esta es la prueba del caso mejor, pues recoge datos de menos parámetros en el intervalo mayor de tiempo propuesto, 10 segundos. En cuestión de ancho de banda, como viene siendo norma, dan valores muy pequeños que indican un consumo bajo (0,000418 Mbps).

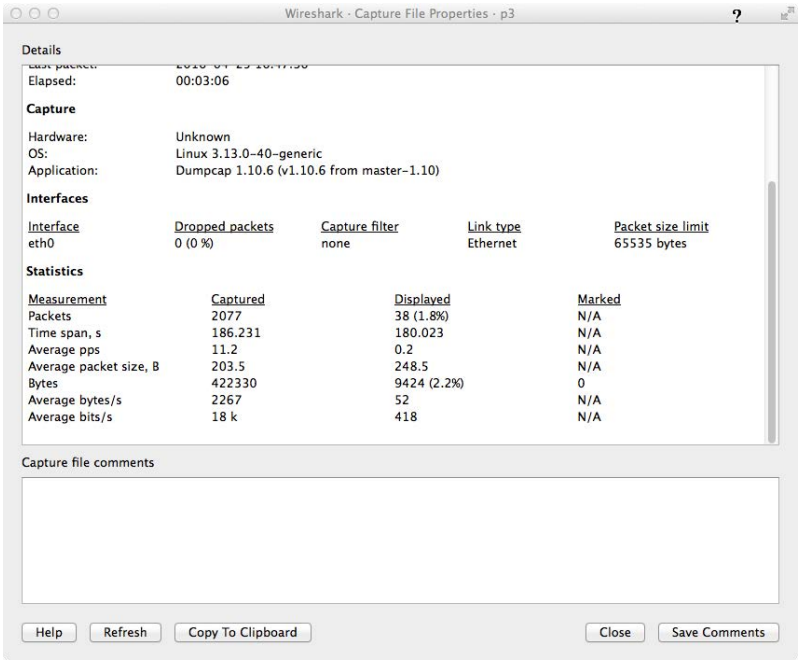


Figura 57. Medición del ancho de banda de la Prueba 3

Además, se puede apreciar que aparte de ser una media baja y razonable, sus valores de consumo de ancho de banda están entre la primera y la segunda prueba, siendo mayor que la primera y menor que la segunda. Esto es importante porque da noción de qué influye principalmente en el consumo de ancho de banda. Teniendo en cuenta las pruebas hasta aquí efectuadas, se puede decir que en su mayor parte, influye el intervalo de tiempo con el que se recogen los datos, y en menor medida, los parámetros que se envían, aunque también es importante este factor. Sin embargo, esto se comprobará una vez acabada la batería de pruebas.

Con respecto al consumo de CPU, también se mantiene pero dando valores inferiores a la segunda prueba e incluso que la primera, demostrando así que la recogida y representación de datos en intervalos de tiempo mayores hace que el consumo de la CPU sea menor, y con ello, que éste es el escenario mejor.

	3			
	user	sys	idle	wa
Media	0,220	0,126	99,602	0,082
Máximo	4,8	1,3	100,0	0,5
Mínimo	0,0	0,0	93,7	0,0

Tabla 9. Datos de consumo de CPU en Prueba 3

Cuarta prueba

La cuarta y última prueba consiste en monitorizar 40 parámetros al azar en intervalos de 10 segundos. En cuestiones de ancho de banda, se obtiene una media de 0.001313 Mbit/sec. Como en anteriores ocasiones, no se considera crítico para el enlace disponible. También confirma las hipótesis anteriores de que el factor más influyente en el consumo de ancho de banda es el intervalo de tiempo, pues como se puede observar, es más cercano a los valores de la primera y tercera prueba que a los de la cuarta. Sin embargo, es también visible el efecto del número de parámetros en cada prueba, lo que quiere decir que aunque un factor parece ser más dominante que otro, ambos son importantes y necesarios a tener en cuenta.

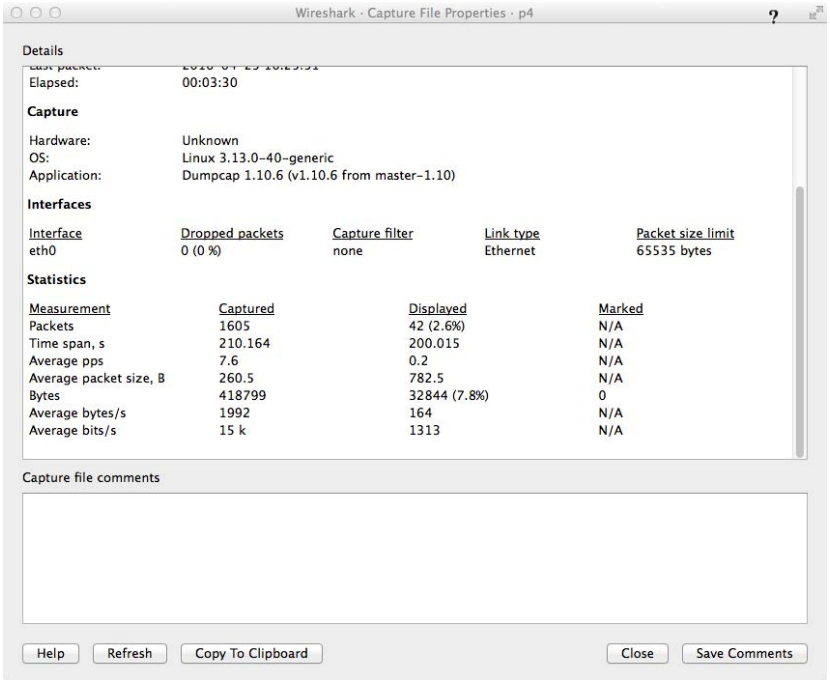


Figura 58. Medición de ancho de banda de la Prueba 4

En temas de consumo de CPU, el resultado es el esperado, estando entre la primera y la segunda prueba, como a priori era lógico. Los valores, como se puede observar, son razonables y parecidos al resto de las pruebas, lo que indica que la cuarta prueba tampoco es crítica en temas de consumo de CPU.

	4			
	user	sys	idle	wa
Media	0,225	0,112	99,586	0,089
Máximo	1,5	0,5	100,0	0,3
Mínimo	0,0	0,0	98,0	0,0

Tabla 10. Datos de consumo de CPU en prueba 4

4.2.2.3 Conclusiones

Como se ha podido observar, el consumo de ancho de banda y de CPU en las distintas pruebas fluctúa en función de los parámetros recogidos y su intervalo de datos. Se han obtenido valores realmente buenos para los propósitos que se quieren alcanzar. De hecho, las pruebas fueron realizadas varias veces cada una para comprobar su efectividad y en todas las ocasiones se lograron resultados parecidos. Si se observan las gráficas de desarrollo se puede ver que en general, el resultado tanto de esta prueba como de las otras es realmente adecuado para el despliegue según sus requisitos.

	1				2				3				4			
	user	sys	idle	wa	user	sys	idle	wa	user	sys	idle	wa	user	sys	idle	wa
Media	0,220	0,152	99,540	0,105	0,225	0,145	99,531	0,115	0,220	0,126	99,602	0,082	0,225	0,112	99,586	0,089
Máximo	4,8	1,8	100,0	0,5	6,5	1,8	100,0	0,5	4,8	1,3	100,0	0,5	1,5	0,5	100,0	0,3
Mínimo	0,0	0,0	93,2	0,0	0,0	0,0	91,4	0,0	0,0	0,0	93,7	0,0	0,0	0,0	98,0	0,0

Tabla 11. Comparativa consumos de CPU

	MONITORIZACIÓN DE ANCHO DE BANDA			
	Prueba 1	Prueba 2	Prueba 3	Prueba 4
MEDIA (Mbps)	0,003986	0,012	0,000418	0,001313

Tabla 12. Comparativa consumos de ancho de banda

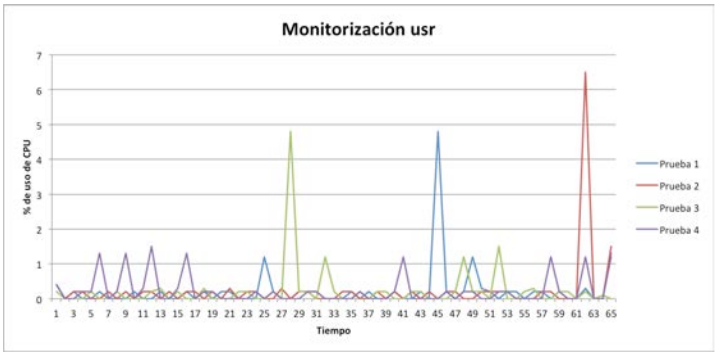


Figura 59. Monitorización de usr (CPU)

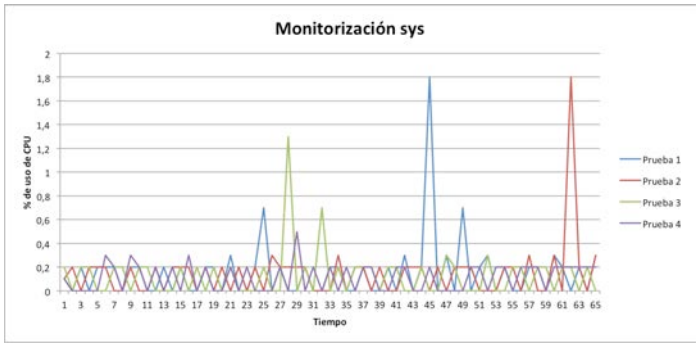


Figura 60. Monitorización de sys (CPU)

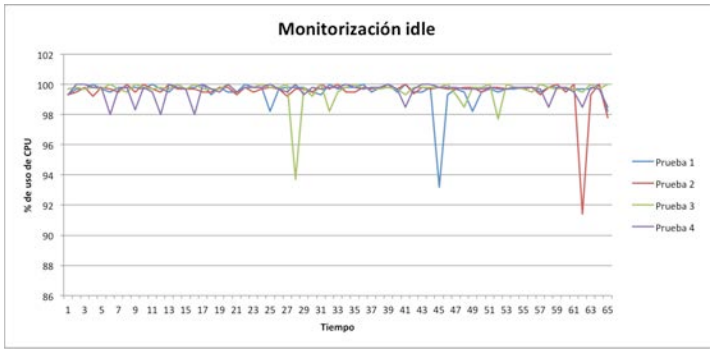


Figura 61. Monitorización de idle (CPU)

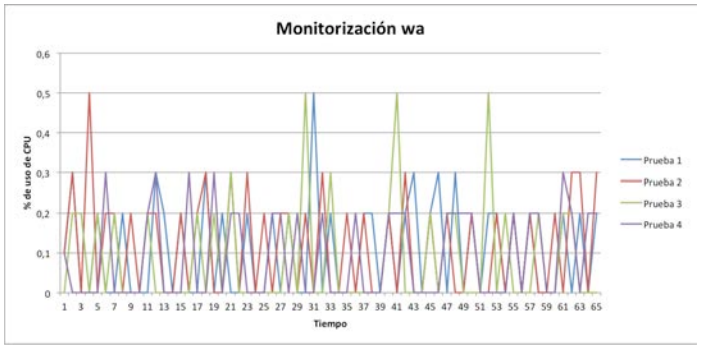


Figura 62. Monitorización de wa (CPU)

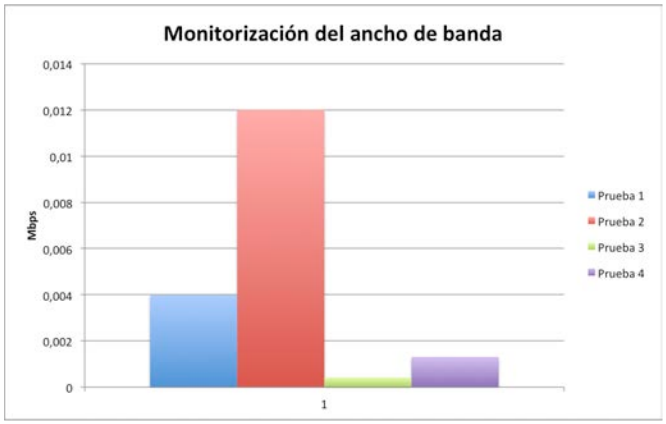


Figura 63. Monitorización del ancho de banda

Como se ha dicho en los apartados anteriores y se puede comprobar, estas variaciones no son críticas para el sistema de red del drone con el router UAV, pues en cuestiones de capacidad, son valores relativamente adecuados y bajos, y no afectan ni al rendimiento de la red ni al del drone.

4.2.3 Pruebas en otros escenarios

4.2.3.1 Máquina virtual

El despliegue sobre una máquina virtual es el primer despliegue adicional en el que se va a probar la herramienta de monitorización Zabbix. Este despliegue es bastante sencillo, pues basta con crear una máquina virtual desde el principio y después instalarle Zabbix según el tipo de distribución elegida para la máquina. Por lo tanto, se ha instalado una máquina virtual usando VirtualBox [Máquina virtual, 2014], con una distribución Ubuntu 14.04.

Con esto, ya se puede arrancar la máquina virtual y usarla sin ningún problema. Como se ha dicho con anterioridad, para instalar Zabbix hay que seguir los mismo pasos que en el modelo terminal portátil – agente monitorizado. En este caso, esta vez se ha instalado la versión 3.0 de Zabbix al tener una distribución de Ubuntu más avanzada.

A partir de aquí, lo único que hay que hacer es configurar Zabbix para que utilizando la herramienta de **importación/exportación** de plantillas podamos recuperar la creada en el primer despliegue y así que la máquina virtual actué como servidor del agente monitorizado que será el router UAV. Para ello, primero se guardan las plantillas tanto de los *hosts* como de las *screens* creadas para el primer despliegue. En cada pantalla de configuración, se eligen los *hosts* y las *screens* y en el botón inferior se selecciona **Import selected**, guardando un fichero XML con cada una de las configuraciones. Una vez hecho esto, se accede a Zabbix de la máquina virtual. En **Configuration - Hosts**, se elige la opción **Import**. En este caso, se ha de importar el fichero XML correspondiente a los *hosts*.

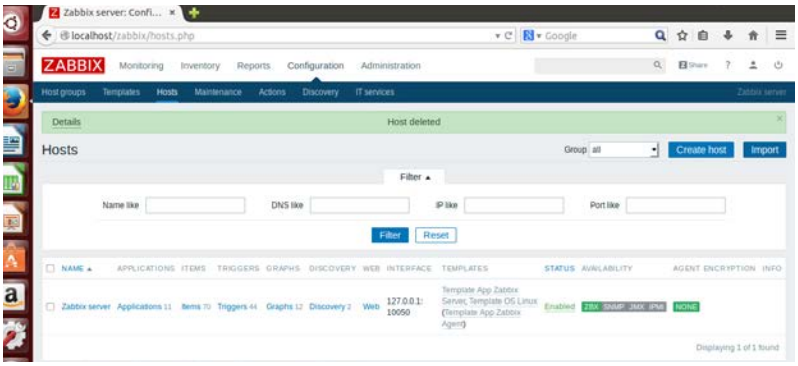


Figura 64. Pantalla *Hosts* en máquina virtual

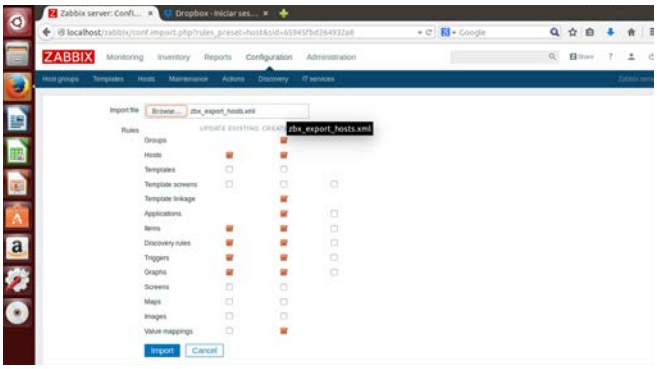


Figura 65. Pantalla importación de templates máquir virtual

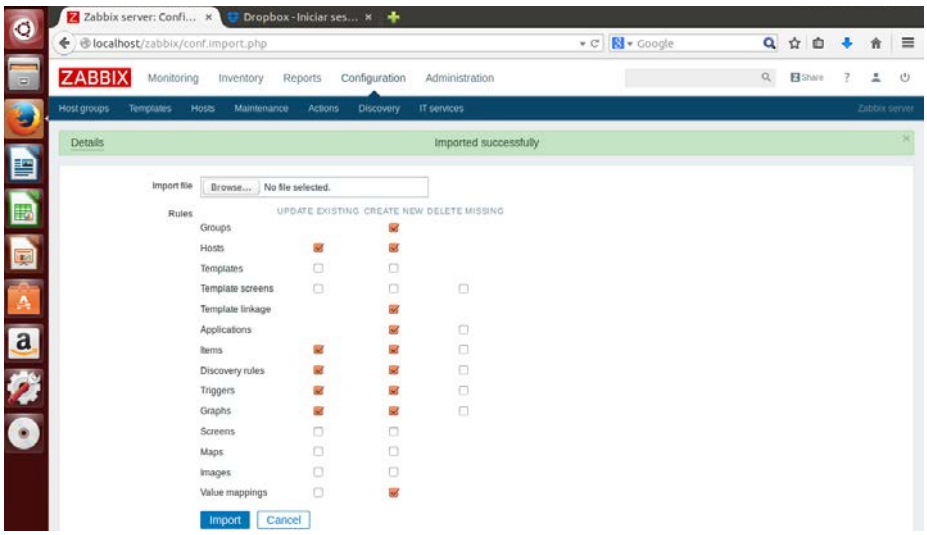


Figura 66. Plantilla instalada satisfactoriamente

Una vez instalada la plantilla de los hosts, se puede ver como aparecen creados los hosts que existían en el primer despliegue satisfactoriamente.

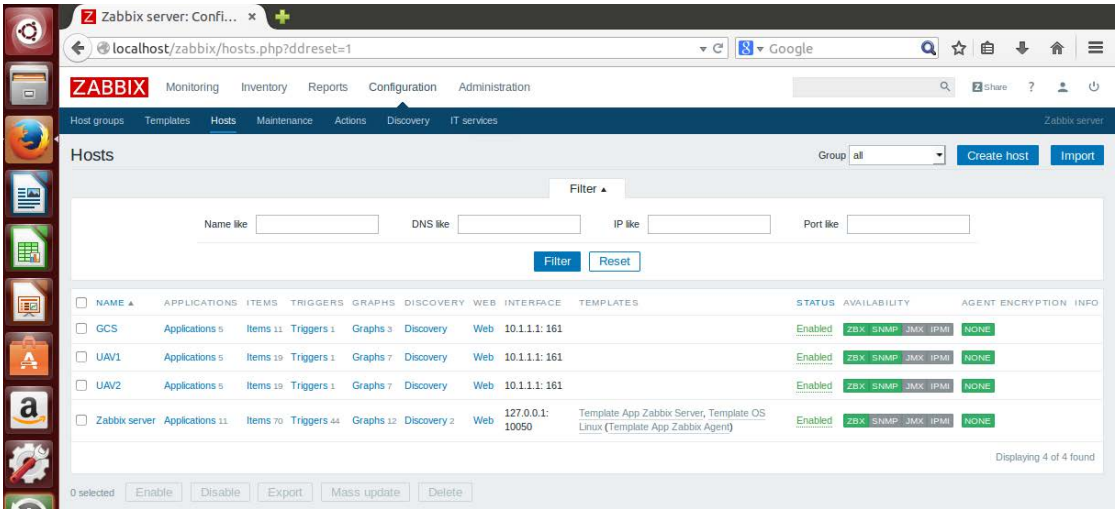


Figura 67. Pantalla Hosts tras la importación de template

Para el caso de las *screens* es el mismo procedimiento. De este modo, una vez instalados todos los templates, esto es lo que se puede ver que hay en la consola Zabbix de la máquina virtual.

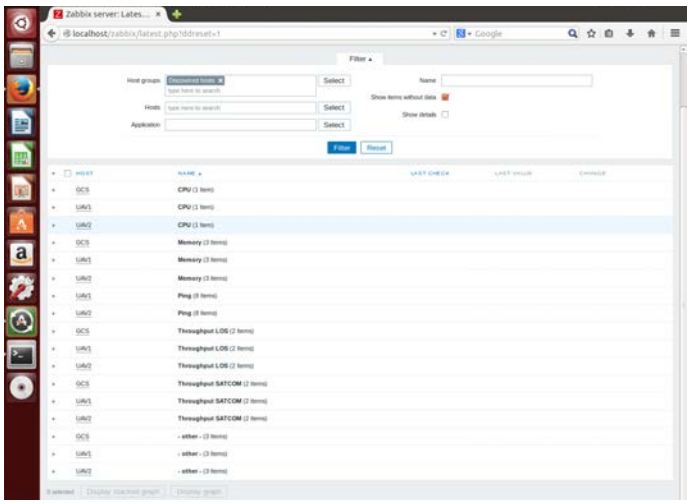


Figura 68. Parámetros monitorizados en Zabbix de máquina virtual

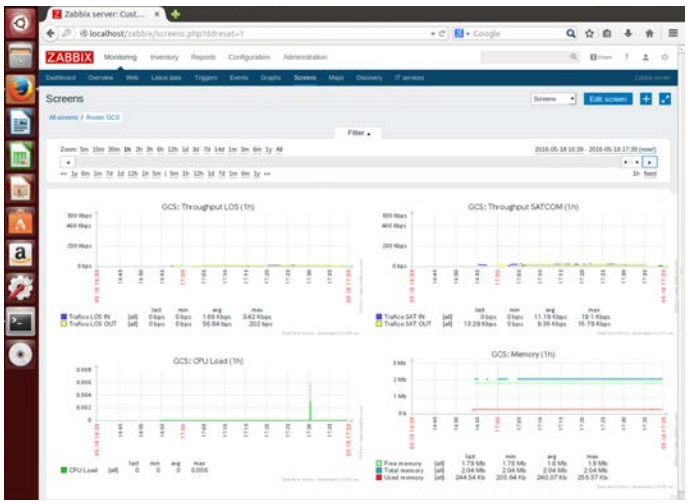


Figura 69. *Screen* router GCS máquina virtual

Figura 70. *Screen* UAV máquina virtual

Para probar que este desarrollo es válido y se puede ejecutar en cualquier dispositivo que sea capaz de gestionar máquinas virtuales, se creará también el fichero **.ova** para pasar la máquina virtual en un pen drive a otro dispositivo y ver que funciona perfectamente.

4.2.3.2 Raspberry Pi

El último despliegue adicional consiste en instalar Zabbix en una Raspberry Pi, de forma que ésta pueda monitorizar cualquier cosa que sea necesaria, funcionando a modo de servidor. Al

igual que en los despliegues anteriores, en primer lugar, se ha de instalar Zabbix en la Raspberry Pi. Posteriormente se procederá a la monitorización de estadísticas de forma que veamos el funcionamiento de Zabbix en este despliegue. Para instalar Zabbix, se ha seguido el siguiente tutorial [Zabbix Raspberry Pi, 2016], modificando algunos pasos según nuestras necesidades. Se puede ver la instalación para el despliegue en el Anexo VI.

Al igual que con la máquina virtual, se importan los *hosts* y las *screens* del primer escenario de manera análoga - previo paso de asegurarnos el alcance al dispositivo monitorizado añadiendo direcciones IP del rango de la subred - demostrando efectivamente, que este escenario también es válido para la monitorización. A continuación, se muestran algunas figuras de este despliegue adicional (figuras 71, 72 y 73).

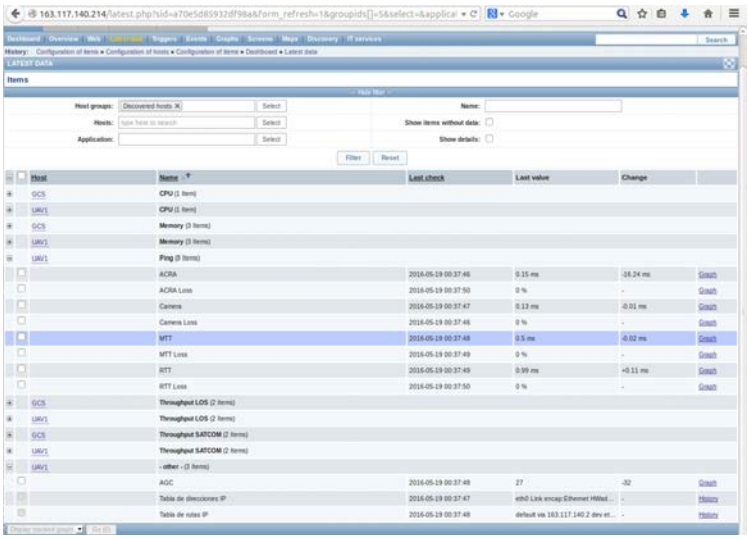


Figura 71. Pantalla de monitorización de parámetros en Raspberry Pi

Figura 72. *Screen* GCS en Raspberry Pi

Figura 73. *Screen* UAV en Raspberry Pi

4.3 Análisis de resultados y evaluación

Como se ha podido comprobar, se han realizado distintas pruebas de evaluación de la herramienta de gestión de red Zabbix y todas ellas han sido ciertamente satisfactorias. No hay que olvidar para que se necesita Zabbix, que es para gestionar y monitorizar la red de un sistema de vehículos aéreos no tripulado y por eso, la validación de las pruebas realizadas es muy importante.

En las pruebas de comprobación de datos e intervalos de SNMP se ha visto que se ha cumplido todo lo estipulado, tanto por el intervalo de recogida de datos como por el cumplimiento del formato de los mensajes SNMP, obteniendo la información adecuada de forma correcta.

Las pruebas de rendimiento eran las pruebas donde se ha comprobado la verdadera actuación de Zabbix y cómo ésta influye según los parámetros e intervalos configurados, obteniendo valores razonables y adecuados dentro de las expectativas.

Finalmente, para probar la flexibilidad de la herramienta, se ha trabajado con las máquinas virtuales y la Raspberry Pi, viendo que efectivamente, Zabbix es una herramienta bastante versátil para la gestión de red.

Se ha comprobado por lo tanto, que gracias a la configuración realizada y las pruebas llevadas a cabo, Zabbix es la herramienta apta para este sistema, cumpliendo todas las expectativas, desde rendimiento hasta flexibilidad en los despliegues.

Capítulo 5

Planificación y presupuesto

5.1 Planificación

La planificación de este proyecto en un principio está pensada para que se cumplimentase en la segunda sesión de defensa del Trabajo de Fin de Grado (julio) dividiendo la planificación en varias fases con duración suficiente para realizar los correspondientes hitos. Para mostrar la planificación, se ha generado el siguiente Diagrama de Gantt, en el que se pueden observar los plazos e intervalos de cumplimiento de los distintos hitos del proyecto [Gantt, 2016].

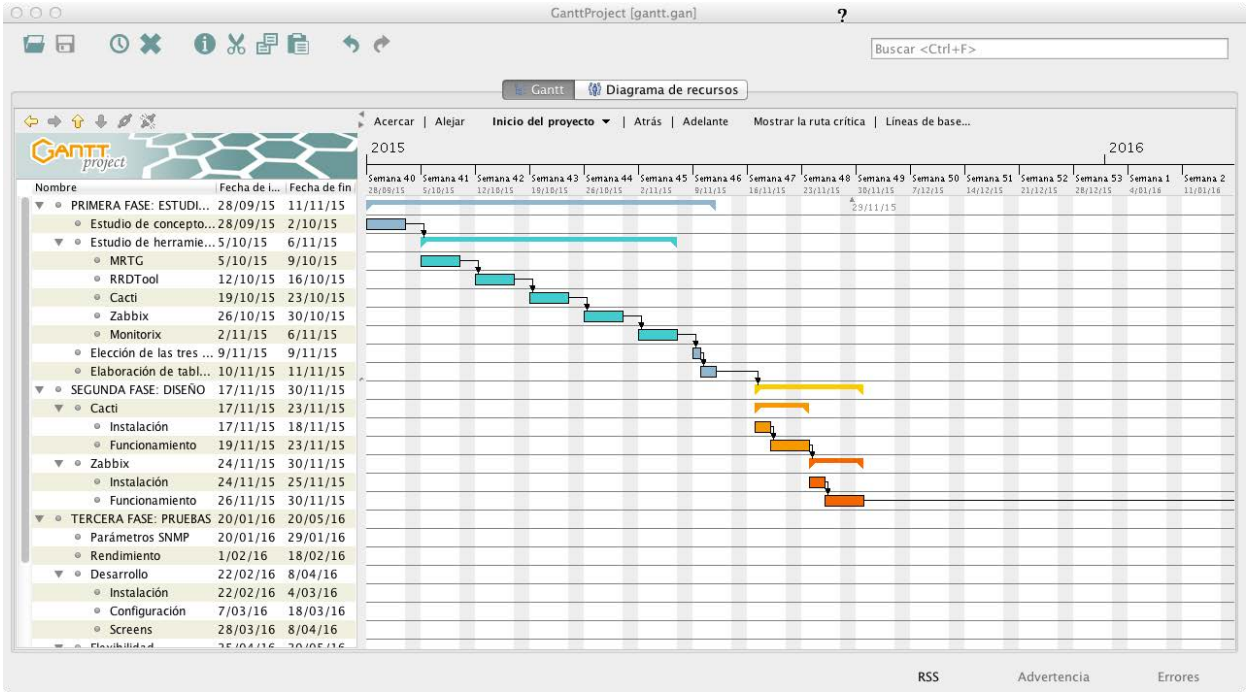


Figura 74. Diagrama de Gantt I

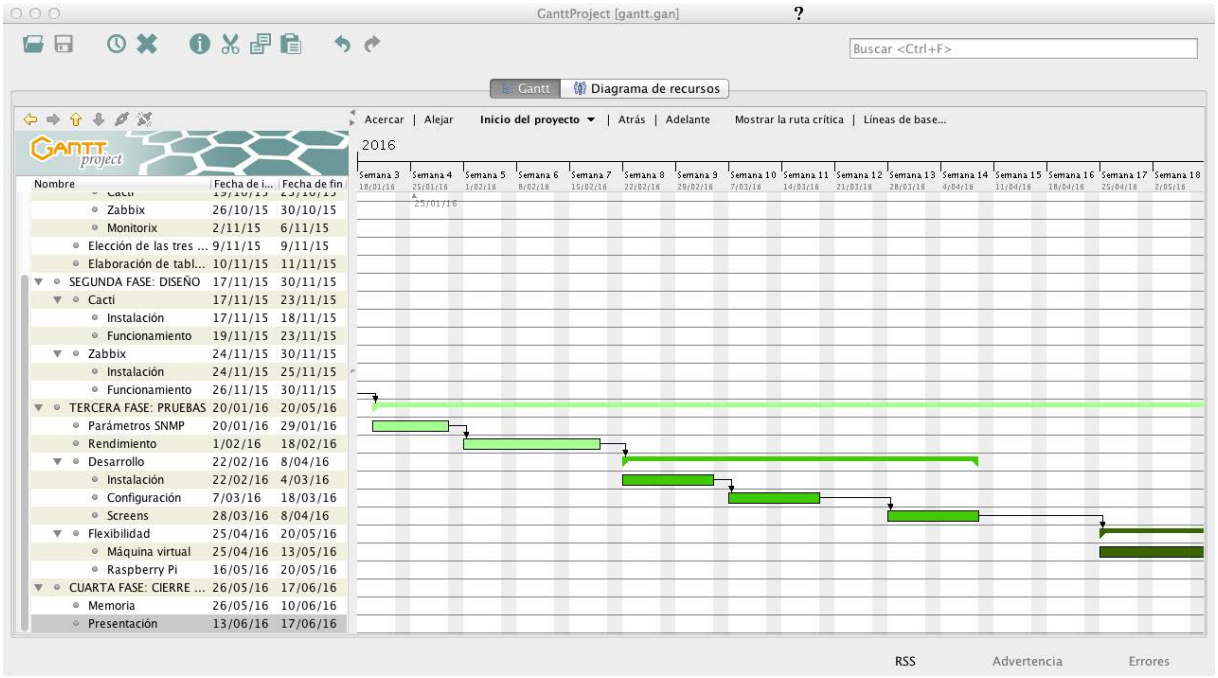


Figura 75. Diagrama de Gantt II

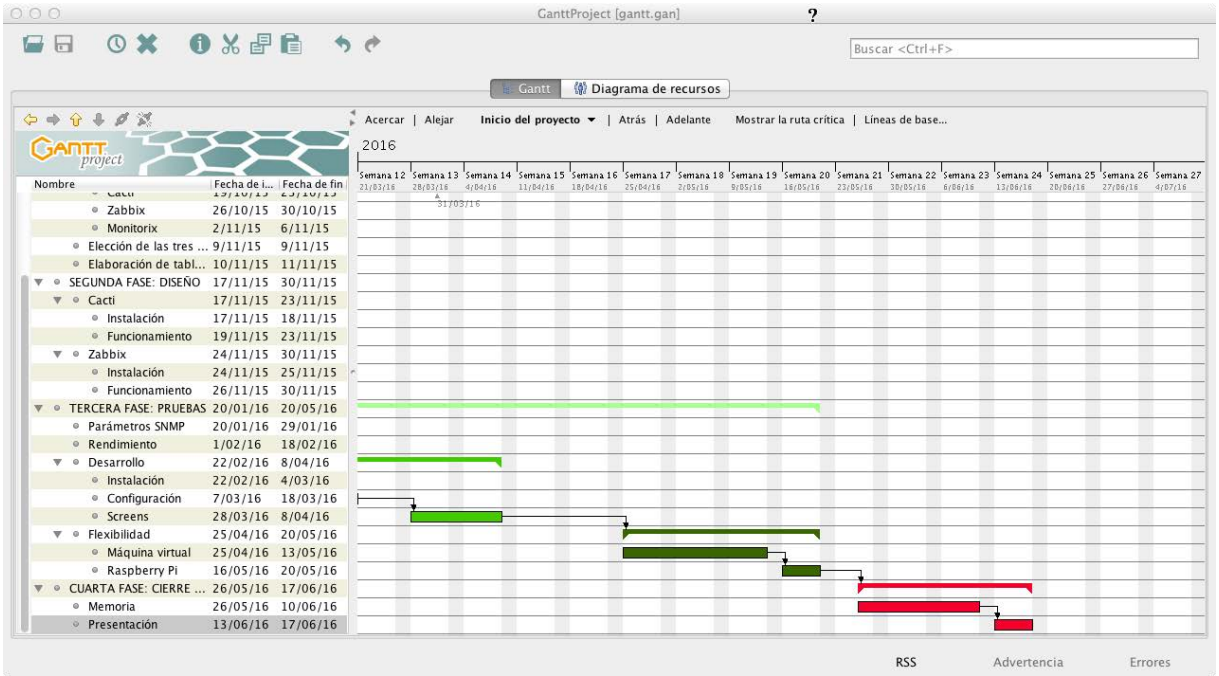


Figura 76. Diagrama de Gantt III

Como se puede observar en el Diagrama de Gantt (partido en tres imágenes porque era muy grande) se ha dividido el proyecto en cuatro fases: estudio previo, diseño, pruebas y desarrollo, y cierre del proyecto. Cada una de estas fases tiene subfases que se cumplen con todos los hitos paso por paso desarrollados en los apartados anteriores con sus duraciones estimadas:

- Estudio previo (azul)
  - Estudio de conceptos de gestión de red

    - Herramientas (MRTG, RRDTool, Cacti, Zabbix, Monitorix)
    - Elección de las tres herramientas elegidas
    - Comparativa de las tres herramientas
- Diseño (naranja)
    - Cacti (Instalación y familiarización con la herramienta a través de un despliegue preliminar)
    - Zabbix (Instalación y familiarización con la herramienta a través de un despliegue preliminar)
- Pruebas y Desarrollo (verde)
    - Pruebas de parámetros SNMP
    - Pruebas de rendimiento
    - Desarrollo (Instalación, configuración y *screens*)
    - Pruebas de versatilidad (Máquina virtual y Raspberry Pi)
- Cierre del proyecto (rojo)
    - Redacción de memoria
    - Creación de presentación

Hay que tener en cuenta que este diagrama es un calendario "aproximado" del desarrollo del proyecto. Se han tenido en cuenta los periodos vacacionales y los fines de semana, en los que los laboratorios no estaban disponibles, dando intervalos de realización ligeramente más largos como extra. También entre fase y fase se ha creado un retraso de aproximadamente 4 días para simular el cambio de contexto de una fase a otra.

5.2 Presupuesto

PRESUPUESTO			
Recurso	**Cantidad u horas empleadas**	**Coste**	**Total coste**
Router UAV	1	- €	- €
Terminal PC	1	400,00 €	400,00 €
Pantalla	1	150,00 €	150,00 €
Raspberry Pi	1	40,00 €	40,00 €
Cable HDMI	1	7,00 €	7,00 €
Ratón con adaptador PS2	1	5,00 €	5,00 €
Ratón con adaptador USB	1	5,00 €	5,00 €
Teclado con adaptador PS2	1	7,00 €	7,00 €
Teclado con adaptador USB	1	10,00 €	10,00 €
Switch	1	75,00 €	75,00 €
Cable de red RJ45	4	6,00 €	24,00 €
Recursos humanos	400	67,85 €	27.140,00 €
TOTAL			**27.863,00 €**

Tabla 13. Presupuesto

El presupuesto se ha realizado teniendo en cuenta costes reales de los recursos empleados de forma aproximada. Las horas de trabajo se han conseguido haciendo una media de las 32 semanas trabajadas con una media de horas por día de 3 horas trabajando 4 días a la semana, donde también se incluyen las horas trabajadas en festivos y fines de semana, dando un total de aproximadamente 400 horas. El coste de la hora trabajada ha sido consultada en la página web oficial del Colegio Oficial de Ingenieros de Telecomunicación [COIT, 2016].

Capítulo 6

Conclusiones

6.1 Conclusiones

Como se ha visto en este Trabajo de Fin de Grado, la implementación de un sistema de gestión de red para vehículos aéreos no tripulados depende de multitud de factores y conceptos que ha sido necesario estudiar y ejecutar para desarrollar este proyecto.

En el apartado Objetivos del Capítulo 1 se pudo comprobar que el objetivo primordial era el diseño y desarrollo de un sistema de gestión de red en un entorno de UAVs, que como se ha visto, ha sido una tarea ardua tras tener que sopesar bastantes parámetros como las prestaciones del diseño, qué parámetros se medirían, el estudio del estado del arte, etc. En ocasiones, surgieron pequeñas dificultades a la hora de tratar de elegir los mejores parámetros de configuración para la herramienta, como en el caso de la elección de los intervalos de recogida de datos, o el modo de recogida de esos datos (a través de agentes SNMP o agentes no SNMP, a través del uso de scripts externos o *User Parameters* …).

Recalcando las conclusiones obtenidas tras las pruebas de validación, se puede decir que se obtuvieron resultados adecuados para el desarrollo del despliegue, como se puede ver a continuación:

- Resultados adecuados en la recogida de datos SNMP. Los datos se recogían en los intervalos adecuados configurados y mantenían el formato de mensajes de SNMP.
- Resultados adecuado de las pruebas de rendimiento y ancho de banda. Los valores obtenidos en dichas pruebas resultaron aptos para las prestaciones del despliegue, demostrando que la recogida de distintas cantidades de parámetros y en distintos intervalos de tiempo no son críticas para afectar al rendimiento ni al ancho de banda.
- Reusltados adecuados de las pruebas de versatilidad. Las pruebas realizadas tanto en despliegues de máquinas virtuales como en plataformas portables de tamaño reducido como la Raspberry Pi dieron resultados aptos para demostrar que herramientas de gestión de red como Zabbix son versátiles en un gran rango de dispositivos.

Otro de los aspectos importantes a tener en cuenta con respecto a las conclusiones es que las herramientas de gestión de datos estudiadas, a pesar de ser versátiles, tienen limitación a la

hora de establecer los intervalos de recogida de datos. Intervalos demasiado pequeños pueden dar lugar a problemas a la hora de representar las gráficos e intervalos demasiado grandes pueden no ser suficientes para la solución que se presenta aquí. Fue por ello importante hacer un despliegue preliminar en el que se probarán dos herramientas para ver cuál de ellas era la que mejor se adaptaba al diseño de la solución.

Desde un punto de vista subjetivo, ha sido un proyecto que ha servido para aprender sobre un campo nuevo, que aunque poco conocido, es primoridal en el correcto funcionamento de las redes de comunicaciones, no sólo adquiriendo nuevos conocimientos, sino también a través de los fallos y aprendiendo de dichos errores. También se ha aprendido sobre como hacer un proyecto, con sus ventajas y desventajas a la hora de organizar, investigar, crear la memoria … A pesar de tener asignaturas en las que te enseñan sobre como hacerlo, nunca se logra bastante soltura hasta que no lo haces sobre un proyecto llevado al completo por uno mismo y además, un proyecto real.

Finalmente, también ha sido importante para abrir la vista a otros conocimientos de otras materias, como los vehículos aéreos no tripulados, dispositivos que según un punto de vista subjetivo, son y serán unos de los inventos tecnológicos más importantes de la sociedad. Nunca hay que dejar de aprender, y si con eso, se consiguen nuevos inventos que consigan hacernos la vida mejor, el ser humano habrá logrado grandes cosas.

6.2 Futuras líneas de trabajo

El proyecto que abarca este Trabajo de Fin de Grado fue presentado como uno de los proyectos relacionados con los vehículos aéreos no tripulados que ofrecía el Departamento de Ingeniería Telemática.

Como tal, puede ser útil para otros proyectos asociados que necesiten de un sistema de gestión de red, no sólo para los dedicados a los vehículos aéreos no tripulados, sino aquéllos que estén relacionados con las redes y su gestión.

Otra futura línea de trabajo en relación al proyecto sería la basada en hacer un desarrollo más amplio del mismo, de forma que se le apliquen más requisitos de diseño según la necesidad establecida, como por ejemplo:

- El uso de *triggers* o alertas en la monitorización de la red, para hacer la vigilancia proactiva más completa. Estas alertas se pueden utilizar para marcar umbrales de uso crítico o también para alertar del mal funcionamiento de algún parámetro.
- Desarrollo de UAVs de tamaño reducido, que actualmente son una línea incipiente de investigación y de desarrollo en el mercado, como hemos mencionado antes, para tareas como rescate, vigilancia, etc.
- Desarrollo de scripts que recojan otros tipos de estadísticas, como sensores acoplados al UAV para medir estadísticas como la temperatura, humedad, velocidad, etc.
- Desarrollo de reglas de auto-descubrimiento para los distintos dispositivos que se deseen monitorizar a través de la red. Esta funcionalidad es muy llamativa si se desea tener redes con muchos dispositivos.

- Creación de *templates* a partir de los parámetros elegidos para el diseño para incorporar posteriormente a otros dispositivos.
- Creación de gráficos más elaborados que incluyan alertas y funciones calculadas con el fin de recoger más datos sobre las estadísticas. Las funciones calculadas servirían para calcular medias, máximos, mínimos, … y después poder representar esos datos en una misma gráfica.

Además, puede servir para los propios trabajos que se han mencionado antes para hacer un buen uso de la red y vigilar proactivamente su actuación, algo que es muy importante en estos despliegues.

Definitivamente, es un tema bastante amplio donde ejercer y crear nuevas ideas. El constante desarrollo en temas de este ámbito puede ser beneficioso para la sociedad y también para la tecnología.

NOTA: Para la lectura del apartado Conclusiones en inglés, se puede consultar el Anexo VII – apartado Conclusions

Anexos

Anexo I. Despliegue preliminar de la red

Paso	Detalles del paso a seguir

1	En primer lugar, se accede al terminal de comandos para hacer unas comprobaciones previas. Se comprueba la configuración IP del ordenador cliente, para ver las interfaces y las subredes a las que está conectado.
	Dentro del terminal:
	`netcom@anemona:~$ ifconfig`
	`eth0 Link encap:Ethernet direcciónHW d8:50:e6:ba:8a:19`
	`Direc. inet:10.1.1.3 Difus.:10.1.1.255 Másc:255.255.255.0`
	`Dirección inet6: fe80::da50:e6ff:feba:8a19/64 Alcance:Enlace`
	`ACTIVO DIFUSIÓN FUNCIONANDO MULTICAST MTU:1500 Métrica:1`
	`Paquetes RX:3423 errores:0 perdidos:0 overruns:0 frame:0`
	`Paquetes TX:178 errores:0 perdidos:0 overruns:0 carrier:0`
	`colisiones:0 long.colaTX:1000`
	`Bytes RX:205380 (205.3 KB) TX bytes:12066 (12.0 KB)`
	`lo Link encap:Bucle local`
	`Direc. inet:127.0.0.1 Másc:255.0.0.0`
	`Dirección inet6: ::1/128 Alcance:Anfitrión`

	ACTIVO BUCLE FUNCIONANDO MTU:65536 Métrica:1 Paquetes RX:192 errores:0 perdidos:0 overruns:0 frame:0 Paquetes TX:192 errores:0 perdidos:0 overruns:0 carrier:0 colisiones:0 long.colaTX:0 Bytes RX:18516 (18.5 KB) TX bytes:18516 (18.5 KB) virbr0 Link encap:Ethernet direcciónHW 0a:15:c5:e0:a7:35 Direc. inet:192.168.122.1 Difus.:192.168.122.255 Másc:255.255.255.0 ACTIVO DIFUSIÓN MULTICAST MTU:1500 Métrica:1 Paquetes RX:0 errores:0 perdidos:0 overruns:0 frame:0 Paquetes TX:0 errores:0 perdidos:0 overruns:0 carrier:0 colisiones:0 long.colaTX:0 Bytes RX:0 (0.0 B) TX bytes:0 (0.0 B)

2	Se añade una dirección IP de la subred a la que está conectada el router UAV al ordenador cliente, de la siguiente forma:
	`netcom@anemona:~$ sudo ip addr add 10.1.2.2/24 dev eth0`
	`[sudo] password for netcom:`
	Como se puede comprobar, exige una contraseña para meter la dirección IP. Es la misma contraseña que la de la cuenta de usuario.

3	Para comprobar que hasta aquí todo es correcto, hay que ver las direcciones IP que actualmente están asignadas a las interfaces distintas que hay en el cliente:
	`netcom@anemona:~$ ip addr show`
	`1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default`
	`link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00`
	`inet 127.0.0.1/8 scope host lo`
	`valid_lft forever preferred_lft forever`
	`inet6 ::1/128 scope host`

	valid_lft forever preferred_lft forever 2: eth1: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN group default qlen 1000 link/ether 00:c0:26:a0:20:a0 brd ff:ff:ff:ff:ff:ff 3: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000 link/ether d8:50:e6:ba:8a:19 brd ff:ff:ff:ff:ff:ff inet 10.1.1.3/24 brd 10.1.1.255 scope global eth0 valid_lft forever preferred_lft forever inet 10.1.2.2/24 scope global eth0 valid_lft forever preferred_lft forever inet6 fe80::da50:e6ff:feba:8a19/64 scope link valid_lft forever preferred_lft forever 4: virbr0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN group default link/ether 0a:15:c5:e0:a7:35 brd ff:ff:ff:ff:ff:ff inet 192.168.122.1/24 brd 192.168.122.255 scope global virbr0 valid_lft forever preferred_lft forever Como se ve en lo marcado en azul, de momento todo está correcto.
4	Otra comprobación que hay que realizar es la de comprobar la conectividad tanto al cliente y al router UAV, el cuál sabemos por una consulta previa dentro de él, que tiene la dirección IP 10.1.2.1: netcom@anemona:~$ ping 10.1.2.1 PING 10.1.2.1 (10.1.2.1) 56(84) bytes of data. 64 bytes from 10.1.2.1: icmp_seq=1 ttl=64 time=0.501 ms 64 bytes from 10.1.2.1: icmp_seq=2 ttl=64 time=0.483 ms ^C --- 10.1.2.1 ping statistics --- 2 packets transmitted, 2 received, 0% packet loss, time 999ms rtt min/avg/max/mdev = 0.483/0.492/0.501/0.009 ms

	`netcom@anemona:~$ ping 10.1.2.2`
	`PING 10.1.2.2 (10.1.2.2) 56(84) bytes of data.`
	`64 bytes from 10.1.2.2: icmp_seq=1 ttl=64 time=0.055 ms`
	`64 bytes from 10.1.2.2: icmp_seq=2 ttl=64 time=0.047 ms`
	`^C`
	`--- 10.1.2.2 ping statistics ---`
	`2 packets transmitted, 2 received, 0% packet loss, time 999ms`
	`rtt min/avg/max/mdev = 0.047/0.051/0.055/0.004 ms`
	Son comprobaciones inanes pero nunca está de más hacerlas.

5	Ahora ya se puede conectar al router UAV a través de SSH, (forma segura) y poder realizar todas las tareas deseadas remotamente. Esto se realiza de la siguiente manera:
	`netcom@anemona:~$ ssh root@10.1.2.1 -p 2222`
	`root@10.1.2.1's password:`
	`Last login: Wed Dec 16 15:27:51 2015 from 10.1.2.2`
	En el despliegue actual, es necesario especificar el puerto 2222 porque por defecto quiere escuchar en el 22, y esto es imposible porque en este despliegue ya está reservado para otras tareas. Nótese que hay que conectarse como *root*, además. Cuando solicita la *password*, hay de introducir la propia del drone (*dronemilano*). Ya se puede acceder al router UAV remotamente.

6	Para finalizar, hay que obtener una dirección IP DHCP (*Dynamic Host Configuration Protocol*) para poder navegar por internet. Tanto en el cliente como en el router (se puede hacer de forma remota), hay que ejecutar el siguiente comando:
	`netcom@anemona:~$ sudo ifconfig eth0 down`
	`netcom@anemona:~$ sudo dhclient eth0`
	`netcom@anemona:~$ sudo ifconfig eth0 up`
	Según la configuración existente, es necesario de “bajar” y “subir” la interfaz antes de ejecutar el comando. Así sabe que en la eth0 tiene que obtener una dirección IP DHCP para poder navegar.

Tabla 14. Pasos a seguir en el despliegue de pruebas

Anexo II. Detalle del despliegue preliminar

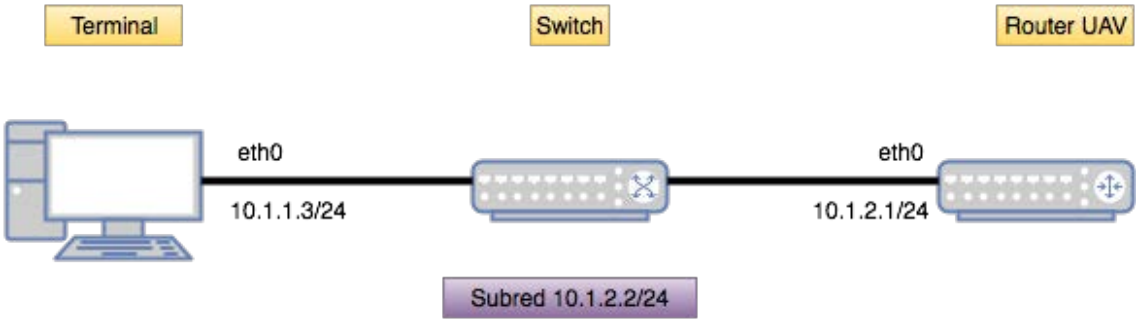


Figura 77. Detalle del despliegue preliminar

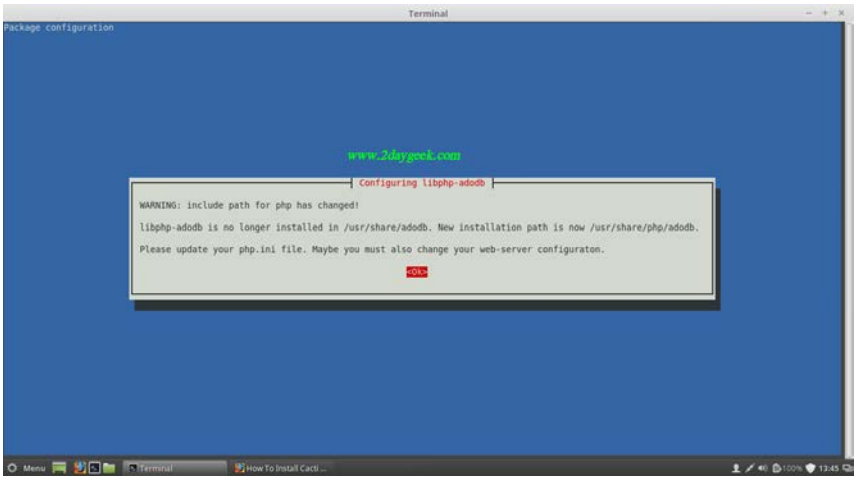
Anexo III. Instalación de las herramientas

Cacti

Paso	Descripción del paso a seguir
1	Después de ejecutar lo establecido en la tabla del despliegue preliminar de la red, se procede a descargar e instalar los paquetes de Cacti con sus correspondientes dependencias. Esto lo hace implícitamente la propia herramienta **apt-get**.

2	Actualizamos repositorio de **apt-get** con el siguiente comando (hemos de hacerlo con permisos de superusuario):
	`netcom@anemona:~$ sudo apt-get update`
	Este comando actualiza el fichero `/etc/apt/sources.list` sincronizando el índice de paquetes desde sus fuentes. Un pequeño vistazo a lo que ocurre al ejecutar el comando es el siguiente:
	`Ign http://dl.google.com stable InRelease`
	`Obj http://dl.google.com stable Release.gpg`
	`Ign http://archive.canonical.com trusty InRelease`
	`[…]`

3	Una vez se ha sincronizado el índice de paquetes, se puede proceder a la instalación. Esto se hace ejecutando el siguiente comando:
	`netcom@anemona:~$ sudo apt-get install cacti`
	Y lo que ocurre es lo siguiente:
	`Leyendo lista de paquetes... Hecho`
	`Creando árbol de dependencias`
	`Leyendo la información de estado... Hecho`

	Se instalarán los siguientes paquetes extras: apache2 apache2-bin apache2-data dbconfig-common fonts-dejavu javascript-common libapache2-mod-php5 libapr1 libaprutil1 libaprutil1-dbd-sqlite3 libaprutil1-ldap libdbd-mysql-perl libdbi-perl […] 0 actualizados, 39 se instalarán, 0 para eliminar y 1 no actualizados. Necesito descargar 18,7 MB de archivos. Se utilizarán 134 MB de espacio de disco adicional después de esta operación. ¿Desea continuar? [S/n] Lo que se ve que ocurre es que para instalar Cacti, son necesarias otras dependencias que como se ha mencionado con anterioridad, que **apt-get** puede instalar a la vez que lo hace con Cacti. Hay que pulsar "S" para continuar.
4	Una vez pulsada "S", se procede a la instalación de todos los paquetes. Durante la instalación de Cacti, aparecerán ventanas de configuración de algunos parámetros que son necesarios. En los siguientes pasos se puede ver cómo hacerlo [Cacti Installation, 2016].
5	Configurar **libphp-adodb**, para acceso universal a todas las bases de datos.  **Figura 78. Configuración libphp-adobd [Cacti Installation, 2016]**
6	Configurar el **servidor web** oportuno. En este caso, se elige **Apache 2**, como en la imagen.

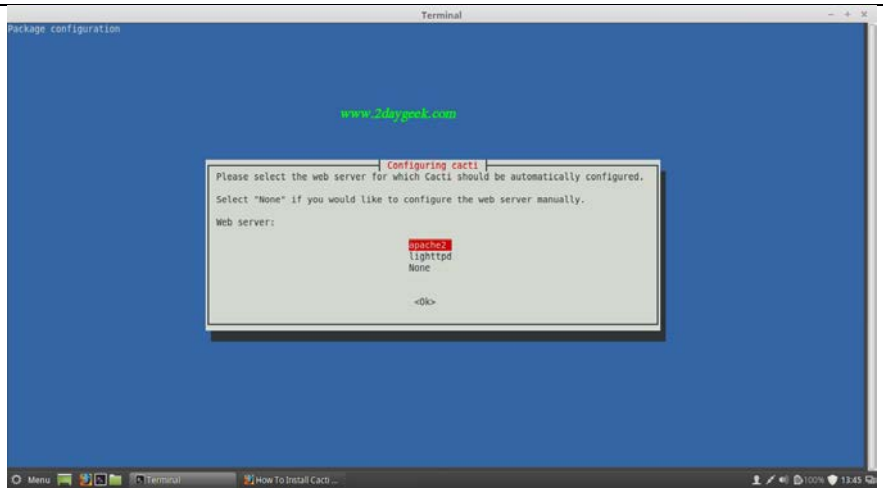
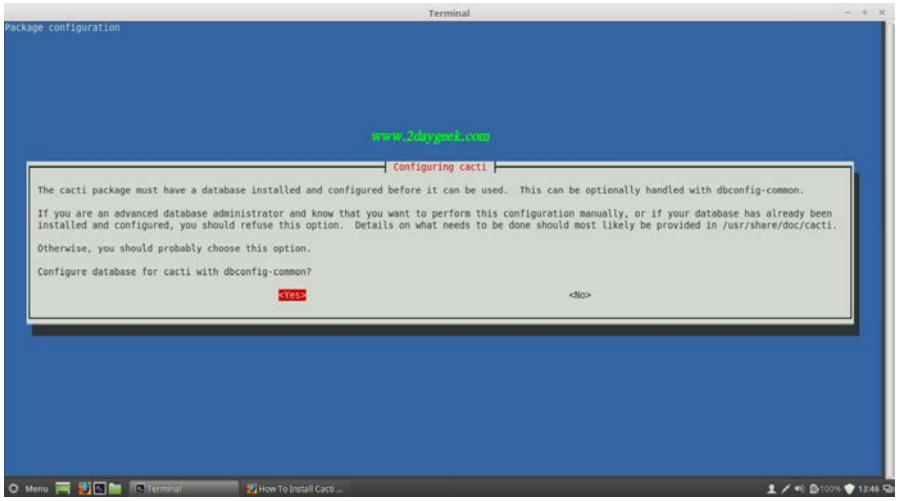
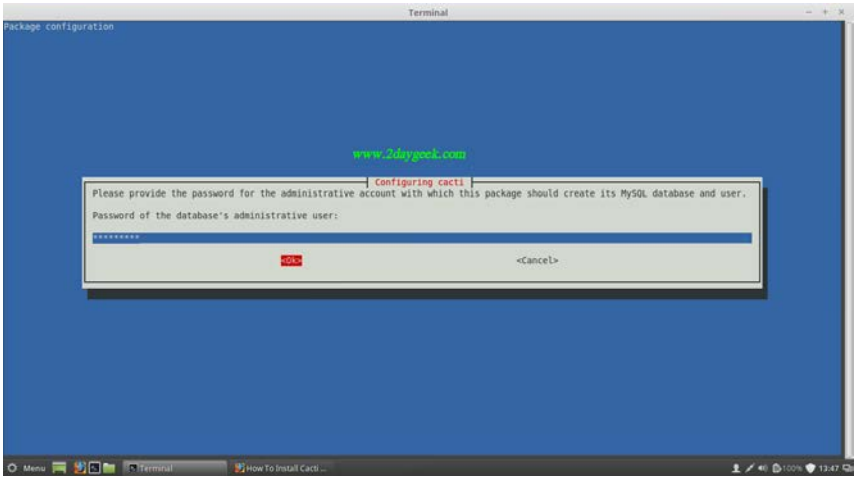
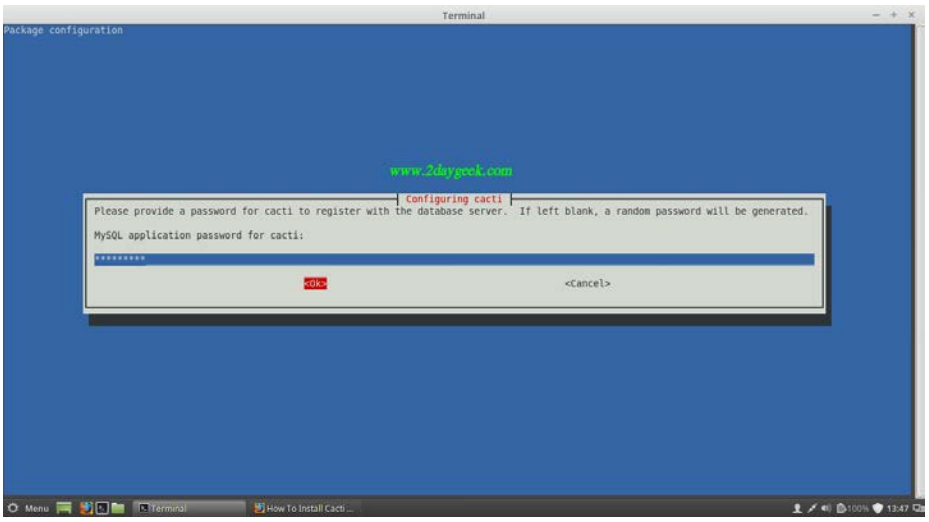
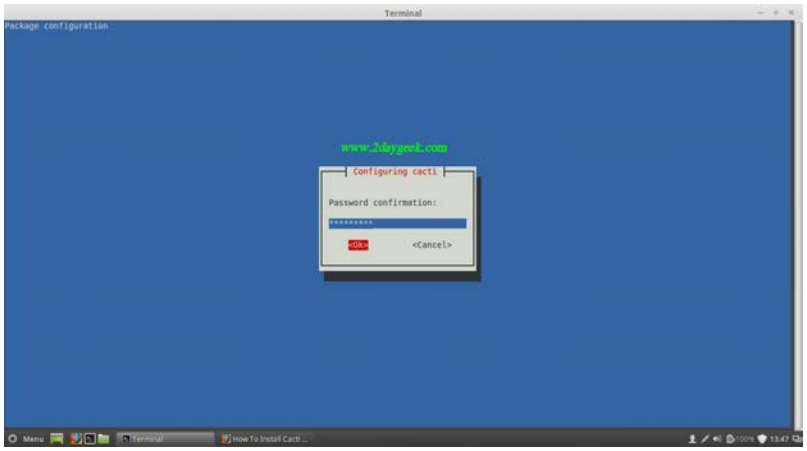
	 **Figura 79. Configuración del servidor web [Cacti Installation, 2016]**
7	Configurar la base de datos de Cacti.  **Figura 80. Configuración de la base de datos de Cacti [Cacti Installation, 2016]**
8	Introducir la *password* de *root* de **MySQL** para Cacti. En este caso, *lucia0109*. 

	Figura 81. *Password* de MySQL para Cacti [Cacti Installation, 2016]
9	Introducir *password* para la propia base de datos de Cacti. Igual que en el paso anterior, *lucia0109*.  **Figura 82. *Password* de base de datos de Cacti [Cacti Installation, 2016]** Después de esto, se confirma contraseña:  **Figura 83. Confirmación *password* de base de datos de Cacti [Cacti Installation, 2016]**
10	Una vez acabado esto, el terminal vuelve al *prompt* normal, dando por finalizado el proceso de instalación por su parte. Ahora hay que configurar el arranque o *setup* de Cacti a través de su interfaz web.
11	Se accede a dicha interfaz web con la siguiente URL, en el caso de monitorización del terminal cliente: http://localhost/cacti (en el caso del router UAV sería http://dirección_IP_router/cacti). Esto redirige a una página de instalación de Cacti:

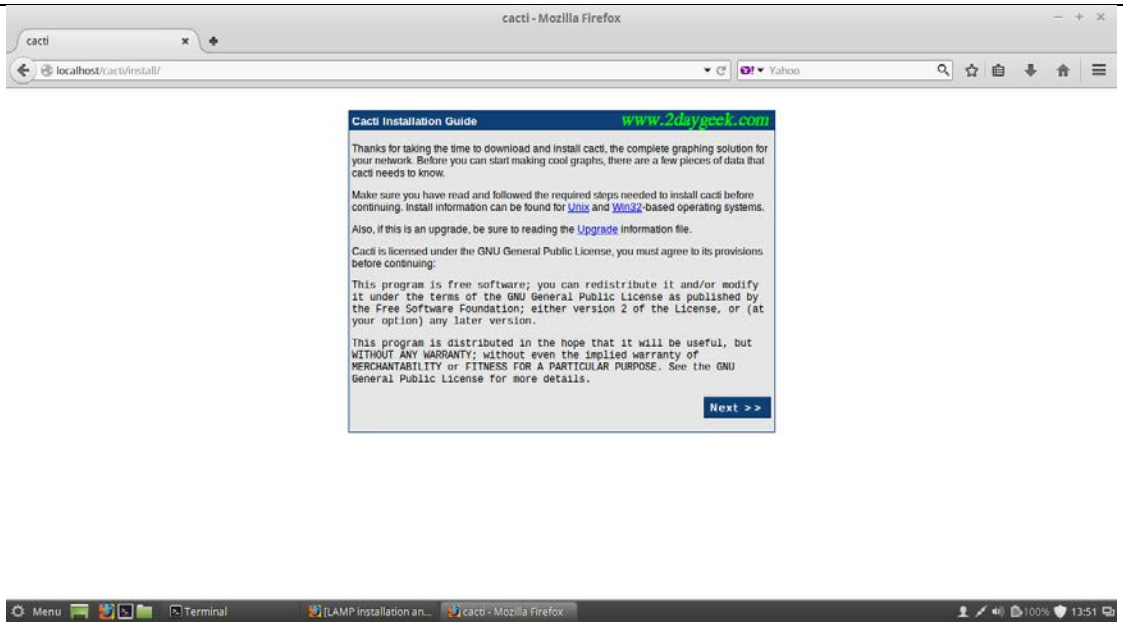


Figura 84. Página principal de instalación de Cacti I [Cacti Installation, 2016]

Se pulsa a *Next*.

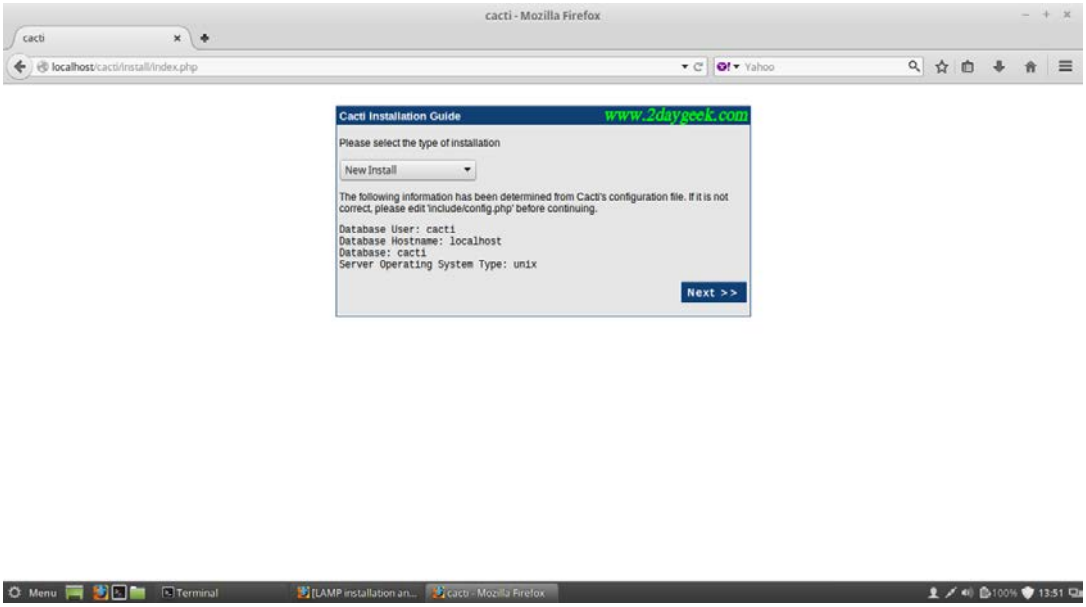


Figura 85. Página principal de instalación de Cacti II [Cacti Installation, 2016]

Por defecto viene *New Installation*. Hay que pulsar a *Next* para continuar.

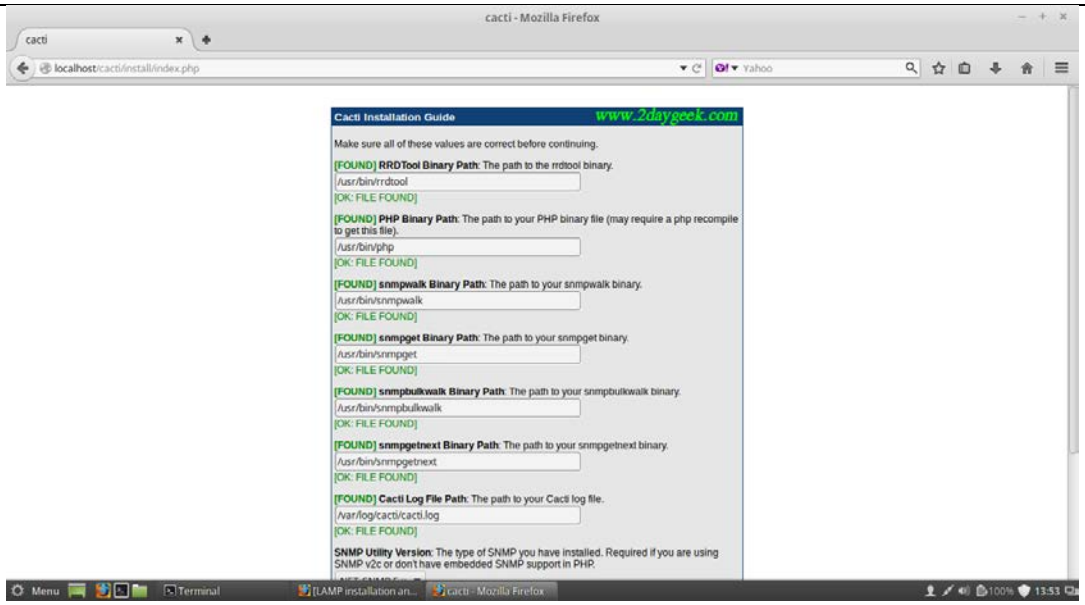


Figura 86. Página principal de instalación de Cacti III [Cacti Installation, 2016]

Cuando esté todo chequeado en verde, la instalación ha sido correcta y se pulsa a *Finish* para finalizar.

12

Una vez instalado, la página redirige al acceso de usuarios.

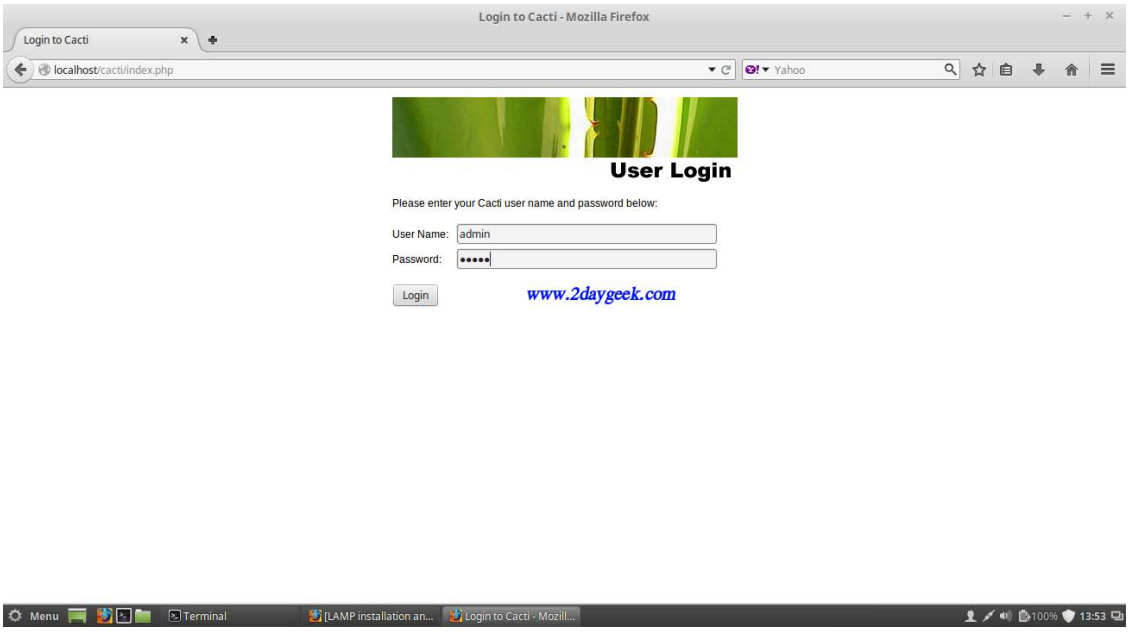


Figura 87. Control de acceso de Cacti [Cacti Installation, 2016]

Por defecto, se puede usar el usuario *admin* con contraseña *admin* para acceder.

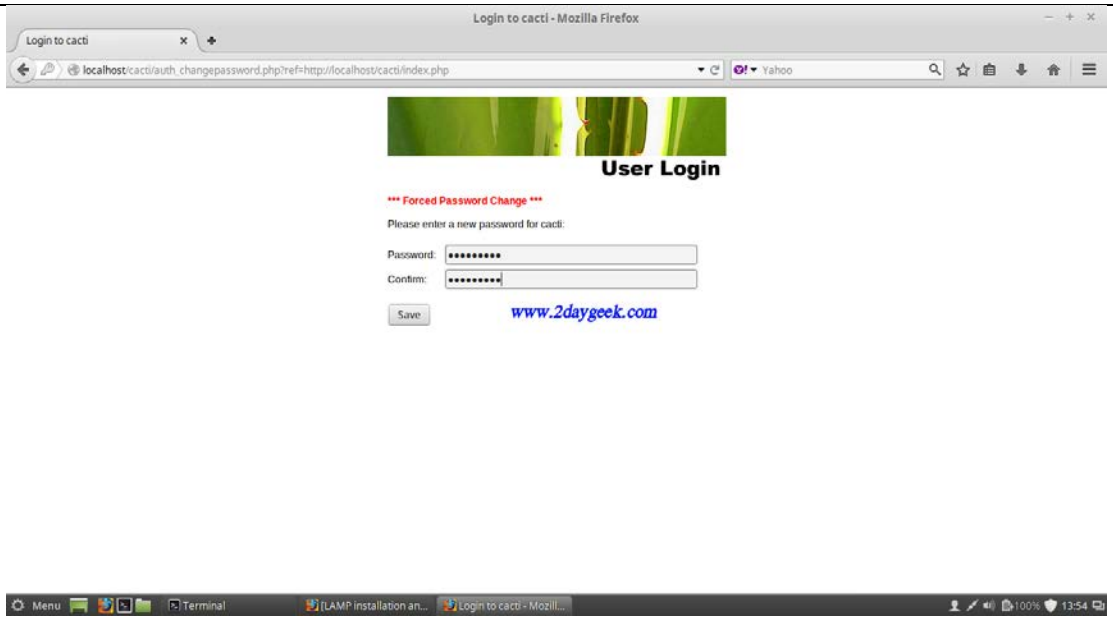
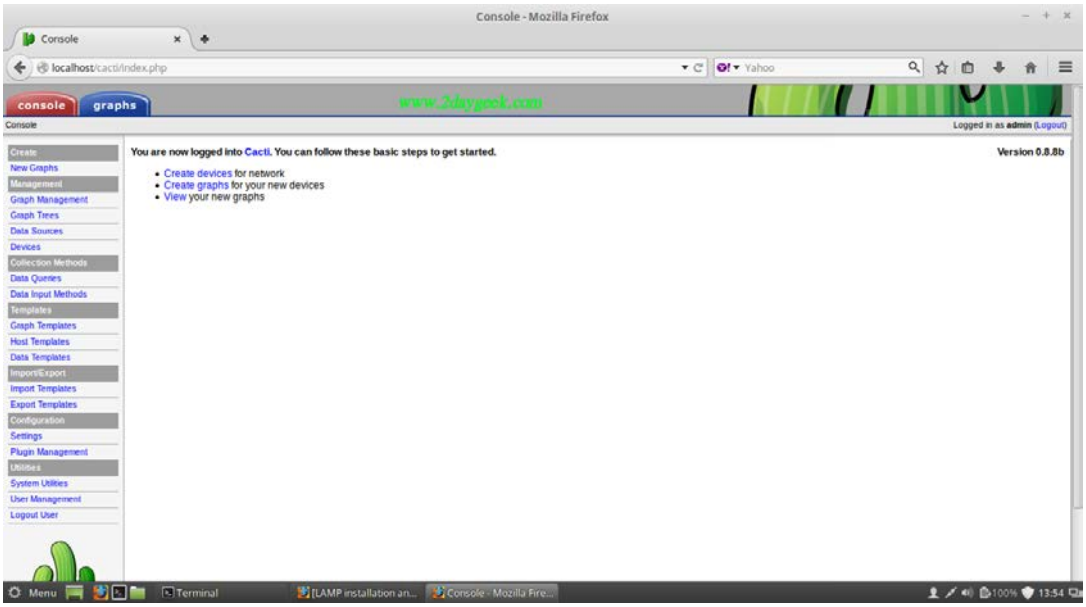
	 **Figura 88. Primer acceso a Cacti [Cacti Installation, 2016]** La primera vez que se acceda, pedirá un cambio de contraseña. La nueva contraseña establecida es *1234*.
13	Una vez finalizado todo, ya se puede acceder a la interfaz de administración de Cacti y a partir de ahí, crear gráficos y monitorizar todos los dispositivos que se deseen.  **Figura 89. Página de inicio de Cacti [Cacti Installation, 2016]**

Tabla 15. Instalación de Cacti

Zabbix

Paso	Descripción del paso a seguir

1	Después de ejecutar lo establecido en la tabla del despliegue preliminar de la red, se procede a descargar e instalar los paquetes de Zabbix con sus correspondientes dependencias. Este proceso es ligeramente distinto a la instalación de Cacti [Zabbix SIA, 2015]. Además, este proceso se va a probar de forma que servidor y agente estén en el mismo dispositivo – el router UAV – por recomendación de Zabbix y porque posteriormente, se probará de forma en que el servidor esté en un terminal y el agente en el router UAV, pero eso será ya en la implementación final. Este primer despliegue es simplemente para ver cómo funciona Zabbix, sus ventajas y desventajas, y si es más oportuno para este proyecto.

2	Se descargan los paquetes Debian pertenecientes a la distribución de Zabbix de la siguiente forma:
	`root@drone-idan-uav:~$ wget http://repo.zabbix.com/zabbix/2.4/debian/pool/main/z/zabbix-release/zabbix-release_2.4-1+wheezy_all.deb`
	A continuación, hay que descomprimir los paquetes descargados con el siguiente comando:
	`root@drone-idan-uav:~$ dpkg -i zabbix-release_2.4-1+wheezy_all.deb`
	Una vez hecho esto, ya están los paquetes necesarios para usar la herramienta **apt-get** e instalar Zabbix y sus dependencias.

3	Al igual que con Cacti, el comando a introducir en este paso es: `root@drone-idan-uav:~$ apt-get update` Esto actualizará el fichero `/etc/apt/sources.list` y ya se pueden instalar los paquetes de Zabbix previamente descargados.
4	Una vez se ha sincronizado el índice de paquetes, según la documentación de Zabbix hay que incluir el siguiente comando, especificando que paquetes de Zabbix queremos instalar en el despliegue. `root@drone-idan-uav:~$ apt-get install zabbix-server-mysql zabbix-frontend-php` Con este comando se empieza a instalar Zabbix. Un fichero de configuración de éste, **dbconfig-common**, configura la base de datos inicial y la llena automáticamente. Como se pudo ver en la instalación de Cacti, se empiezan a instalar las dependencias necesarias y hay que seguir los pasos que sean oportunos y que pida la instalación.
5	Durante la instalación de Zabbix, aparecerán ventanas de configuración de algunos parámetros que son necesarios, al igual que con Cacti. En los siguientes pasos se pueden observar las imágenes de dichas configuraciones.
6	Configuración de **zabbix-server-mysql**.

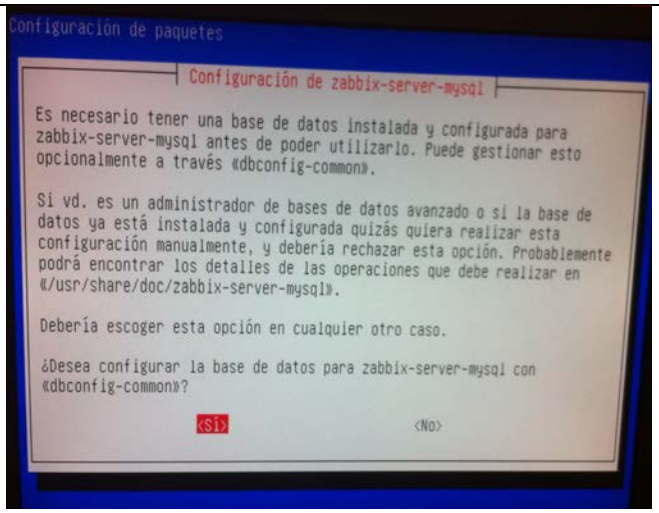
<table>
<tr><td></td><td> 

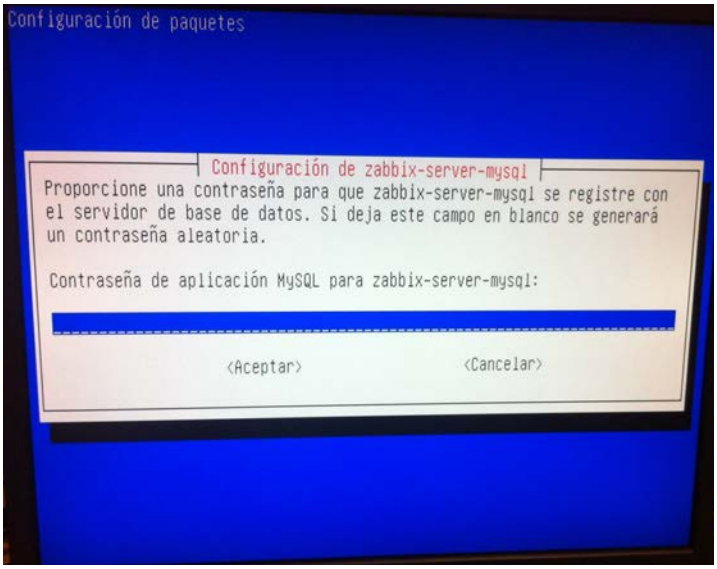
Figura 90. Configuración de zabbix-server-mysql para Zabbix</td></tr>
<tr><td>7</td><td>A esta configuración de zabbix-server-mysql hay que completarla con una contraseña, lucia0109.
 

Figura 91. Contraseña para zabbix-server-mysql para Zabbix
Después de escribirla se pide confirmación:</td></tr>
</table>

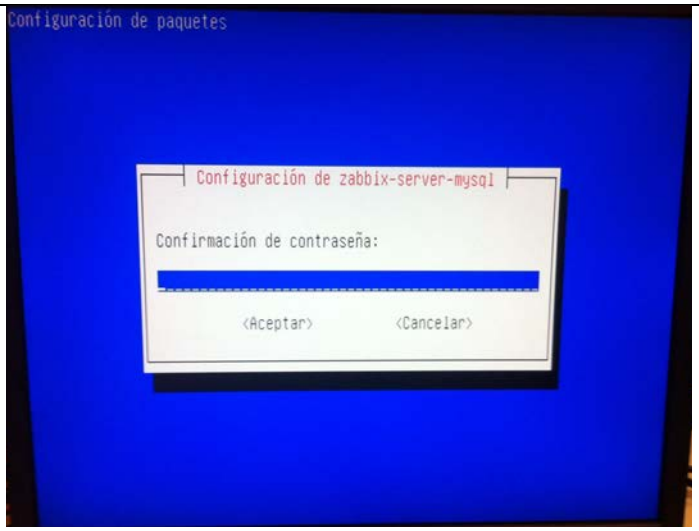


Figura 92. Confirmación de contraseña para zabbix-server-mysql para Zabbix

8 También hay que configurar **mysql-server-5.5**. Aunque no sea necesario, se pide que incluya una contraseña para el usuario *root* de MySQL. Al igual que siempre, se elige *lucia0109*.

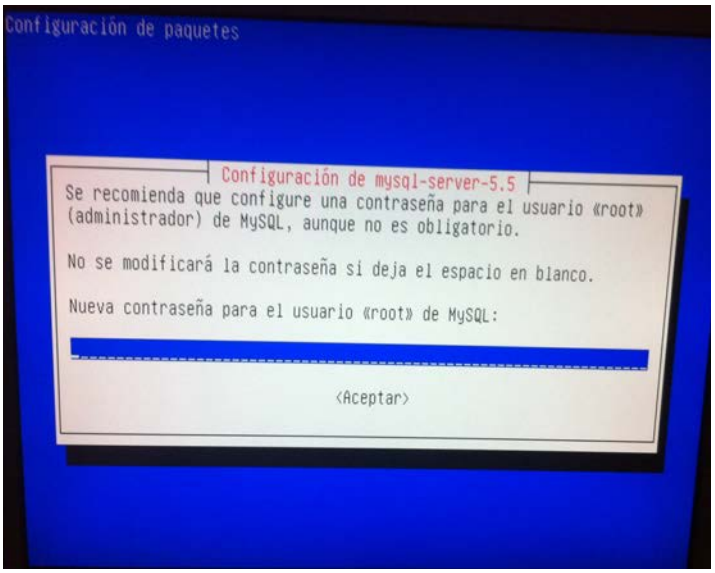


Figura 93. Configuración de mysql-server-5.5 para Zabbix

De inmediato se exige confirmación de dicha contraseña:

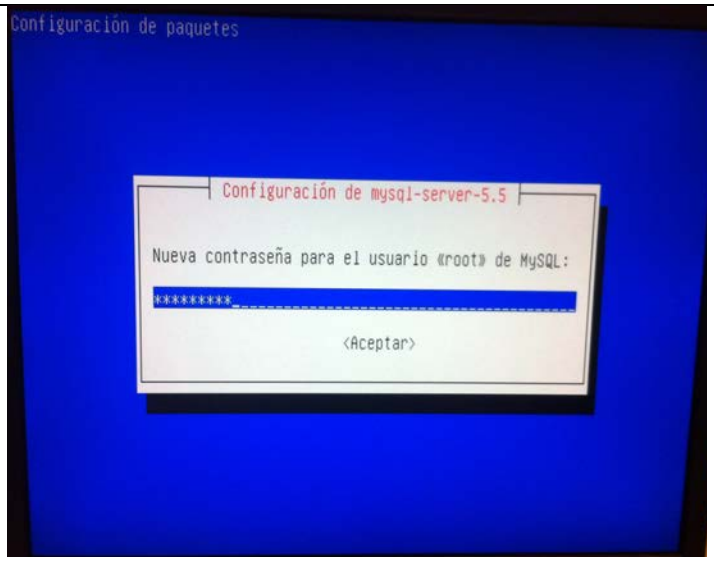


Figura 94. Confirmación de contraseña mysql-server-5.5 para Zabbix

9 Para que la configuración PHP de Zabbix sea correcta, hay que cambiar el fichero de configuración existente, disponible en `/etc/apache2/conf.d/zabbix`. Éste tiene una forma como la siguiente:

```
php_value max_execution_time 300
php_value memory_limit 128M
php_value post_max_size 16M
php_value upload_max_filesize 2M
php_value max_input_time 300
# php_value date.timezone Europe/Riga
```

Para que funcione bien, hay que cambiar la línea comentada y poner una zona horaria correcta [PHP, 2016]. Con el editor de ficheros **vi** se cambia la zona horaria a Madrid. En este caso, el fichero queda al final de la siguiente forma:

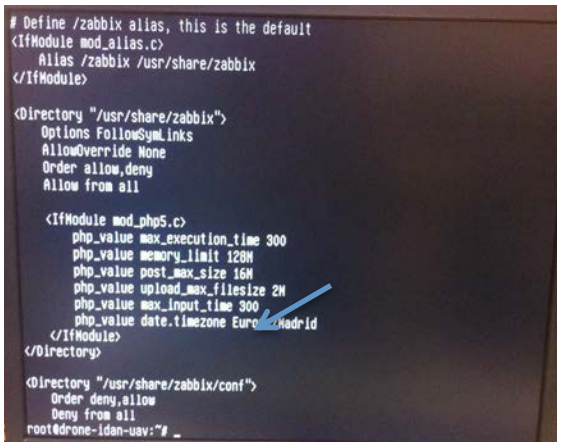


Figura 95. Fichero /etc/apache2/conf.d/zabbix después de modificación

Para que funcione, ahora es necesario reiniciar el servidor Apache con el siguiente comando:

```
root@drone-idan-uav:~$ service apache2 restart
```

	Con esto, ya se puede acceder a la configuración inicial de Zabbix en la aplicación Web.
10	Además, si se desea instalar un agente Zabbix en la misma máquina que actúa de servidor (despliegue recomendado), hay que introducir el siguiente comando: `root@drone-idan-uav:~$ apt-get install zabbix-agent` Como ya se han instalado las dependencias de Zabbix al instalar el servidor, no es necesario hacerlo cuando se instala el agente.

Tabla 16. Instalación de Zabbix

Anexo IV. Scripts

Valor AGC	`# Script que simula el comportamiento de volcado de valores AGC sobre el router UAV`
	
	`echo $(($RANDOM%100))`

Ping *Loss*	`# Script que recoge las estadísticas que se obtienen con un ping, en concreto la pérdida de paquetes`
	
	`# Variables a poner en la línea de comandos. La $1 es la dirección IP a la que se hace ping`
	
	`ip=$1`
	
	`# packet loss`
	`ping –c 5 $ip|grep packet|awk '{print $6}'|cut –d'%' –f1`

Ping RTT	`# Script que recoge las estadísticas que se obtienen con un ping, en concreto el RTT medio`
	
	`# Variables a poner en la línea de comandos. La $1 es la dirección IP a la que se hace ping`
	
	`ip=$1`

	`# RTT avg` `ping -c 5 $ip|grep rtt|awk '{print $4}'|cut -f1| awk '{split($0,a,"/"); print a[2]}'`
Tabla de direcciones IP	`# Script que muestra la tabla de direcciones IP. Se ejecuta sobre el agente` `ifconfig`
Tabla de rutas IP	`# Script que muestra la tabla de rutas IP. Se ejecuta sobre el agente` `ip ro`

Tabla 17. Scripts

Anexo V. Lista de OID's utilizados en el Desarrollo

Parámetros Zabbix	**OID**
Memoria	.1.3.6.1.4.1.2021.4.6.0 – *Free memory* (en la web viene como *Total RAM used*, pero es una errata) .1.3.6.1.4.1.2021.4.5.0 – *Total memory* Para *Used memory*, el OID de la web no servía, pero se calculó como una resta entre las dos anteriores.
***Throughput* SATCOM**	.1.3.6.1.2.1.2.2.1.16.2 – *Throughput OUT* .1.3.6.1.2.1.2.2.1.10.2 – *Throughput* IN Estos OID en realidad dan los resultados de los octetos que entran/salen de la interfaz. Para medir el *throughput*, se usa la función de los ítems Zabbix para guardar el valor como *Delta (speed per second).*
***Throughput* LOS**	.1.3.6.1.2.1.2.2.1.16.4 – *Throughput* OUT .1.3.6.1.2.1.2.2.1.10.4 – *Throughput* IN

Tabla 18. Lista de OID's utilizados

Anexo VI. Instalación de Zabbix en Raspberry Pi

1. Instalación de paquetes necesarios:

```
$ sudo apt-get install make gcc libc6-dev libmysqlclient-dev libcurl4-openssl-dev libssh2-1-dev libsnmp-dev libiksemel-dev mysql-server libopenipmi-dev fping php5-gd snmp libsnmp-base openjdk-6-jdk unixodbc unixodbc-dev libxml2 libxml2-dev snmp-mibs-downloader snmpd python-pywbem php5-ldap php5-mysql
```

2. Aparecerá después la pantalla para asignarle la contraseña al usuario root de MySQL. Se elige la contraseña *lucia0109*.
3. Una vez instalados los paquetes dependientes, se descargan y se desempaquetan los paquetes Zabbix, descargados de la web, en la carpeta creada para Zabbix.

```
$ sudo /usr/src
$ sudo wget
http://downloads.sourceforge.net/project/zabbix/ZABBIX%20Latest%20Stable/2.4.4/zabbix-2.4.4.tar.gz
$ sudo tar -xzvf zabbix-2.4.4.tar.gz
```

4. Una vez descargado y descomprimido, hay que acceder al directorio creado – Zabbix-2.4.4. –y se introduce el siguiente script de configuración.

```
$ sudo ./configure --enable-server --enable-agent --with-mysql --with-net-snmp --with-libcurl --with-openipmi --with-ssh2 --with-libxml2 --enable-ipv6 --enable-java --with-jabber --with-unixodbc --with-ldap
```

5. Se compila y se instala con *make* y *make install*. Esto puede tardar varios minutos.

```
$ sudo make
$ sudo make install
```

6. Cuando esté todo instalado ya, el siguiente paso será crear el usuario "zabbixrasppi". Este se encarga de la ejecución y de las secuencias de comandos del entorno "Zabbix". Con el siguiente comando se crea el usuario con una "UID" de sistema, se deniega el inicio de sesión interactivo y dirige su directorio principal al directorio de la instalación Zabbix:

```
$ sudo adduser --system --home /usr/local/sbin --no-create-home zabbix
```

7. Se accede a la base de datos MYSQL como *root*:

```
$ mysql –u root -p
```

8. Se crea una base de datos con el nombre de "zabbix" y se otorgan los permisos necesarios para que el usuario "zabbix" pueda controlar la base de datos creada desde nuestra Raspberry

```
>create database zabbix;
>grant all on zabbix.* to zabbix@localhost identified by 'zabbix';
```

9. Una vez creada la base de datos, se sale de MySQL y se importan las plantillas, para construir las tablas y poblar la base de datos:

```
$ sudo mysql -u zabbix --password=zabbix zabbix </usr/src/zabbix-2.4.4/database/mysql/schema.sql
$ sudo mysql -u zabbix --password=zabbix zabbix </usr/src/zabbix-2.4.4/database/mysql/images.sql
$ sudo mysql -u zabbix --password=zabbix zabbix </usr/src/zabbix-2.4.4/database/mysql/data.sql
```

10. Ahora se harán algunos modificaciones en el fichero de configuración de Zabbix.

```
DBHost=localhost # Servidor de la base de datos.
DBName=zabbix # Nombre de la base de datos.
DBUser=zabbix # Usuario de la base de datos
DBPassword=zabbix # Contraseña de la base de datos
Timeout=30 # Tiempo en segundos que tiene que esperar el agente.
ExternalScripts=/usr/local/share/zabbix/externalscripts # Archivo para los scripts externos.
AlertScriptsPath=/usr/local/share/zabbix/alertscripts # Archivo para los scripts de alertas.
FpingLocation=/usr/bin/fping # Directorio del binario "Fping" para comprobar la conectividad de los "hosts".
```

11. Es importante que Zabbix se inicie automáticamente en el arranque del sistema. Para ello se copiarán los scripts de inicio de Zabbix en el directorio `/etc/init.d`.

```
$ sudo cp /usr/src/zabbix-2.4.4/misc/init.d/debian/zabbix-* /etc/init.d/
```

12. Posteriormente se abrirá el archivo `/etc/rc.local` y se añadirán los scripts para que el servidor y el agente se ejecuten al iniciar el sistema.

```
$ sudo vi /etc/rc.local
```

Se insertan estas líneas:

```
/etc/init.d/zabbix-server start
/etc/init.d/zabbix-agent start
```

13. Se reinicia la Raspberry Pi y se ejecuta el siguiente comando para comprobar que los servicios se han iniciado correctamente.

```
$ sudo ps aux | grep zabbix
```

14. Lo último que falta configurar es el *Frontend*. Como se ha visto con anterioridad, Zabbix se apoya en la interfaz web con la que el usuario puede administrar fácilmente el servidor. Para configurar **apache2**, se copiarán los archivos necesarios en la carpeta que por defecto utiliza dicho programa.

```
$ sudo cp -R /usr/src/zabbix-2.4.4/frontends/php/* /var/www
```

15. Ahora se asignarán los permisos de los directorios y archivos para el usuario y grupo "www-data", y se elimirá el archivo "index.html" del servidor Web.

```
$ chown -R www-data:www-data /var/www/*
$ sudo rm /var/www/index.html
```

16. Para que todo funcione correctamente es neceario cambiar ciertos parámetros de "PHP" que por defecto son insuficientes para que el servicio funcione correctamente. Se edita el archivo "php.ini" con el siguiente comando:

```
$ sudo vi /etc/php5/apache2filter/php.ini
```

Y se modifican los siguientes parámetros:

```
post_max_size = 16M
max_execution_time = 600
max_input_time = 600
date.timezone = Europe/Madrid
```

17. Para que estos ajustes tengan efecto, se reinicia **apache2**.

```
$ sudo apache2ctl restart
```

18. Ahora se procederá a la configuración del *Frontend*. Para ello, se accede a él mediante la IP o con el nombre de nuestro servidor con el navegador que deseemos.

```
http://ip_o_nombre_de_la_raspberry/
```

19. Se mostrará un asistente de configuración que tantas veces hemos modificado. Se siguen los pasos y al llegar al final ya se tiene Zabbix instalado en la Raspberry Pi.

Anexo VII. English competence

Introduction

Motivation

In these days, we are very used to the concept of “network”. It could be the local network at home, or the mobile one someone can connect to when they go out, the fact is that networks are always present in our lives.

Taking this into account, it is normal for us to think of better things to do with networks. Imagine a lot of dangerous situations such as a car accident, a climber lost in the mountain, an earthquake, … Nowadays, the arrival time of help is reduced, but it could be better. With the advance of new technologies as well as the development of networks, it could be possible to improve it thanks to the proactive network monitoring. Here is where the UAV (*Unmanned Aerial Vehicle*) and the concept of network comes into view.

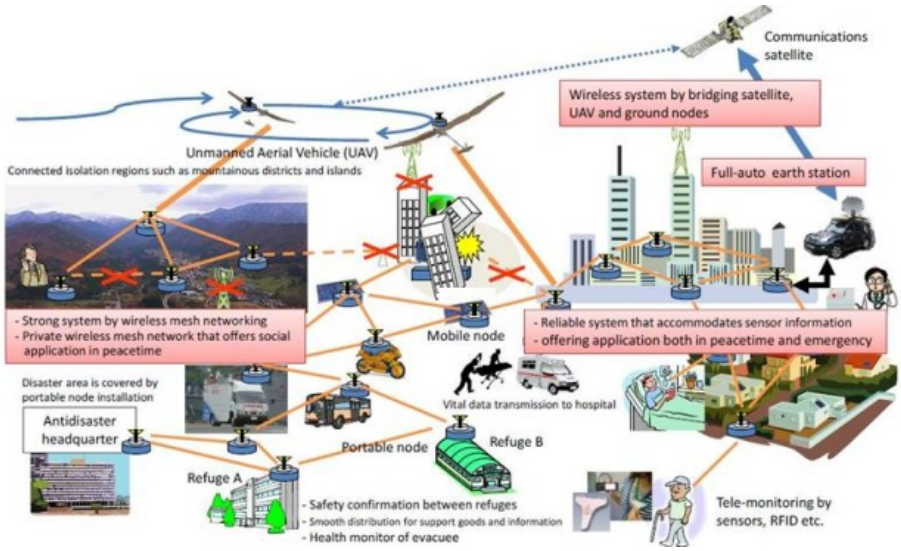


Figura 96. Mesh network with UAV, satellite and ground nodes [UAV Network, 2016]

With an UAV, as well as with other network components, huge mesh networks could be developed in order to cover and watch some areas, for instance, areas prone to earthquakes. This is useful to avoid the dangerous situations proactively or when the disaster occurs, solve it and help the potential affected people as soon as possible [UAV Network, 2016].

On the other hand, it is normal that networks have occasional problems, due to their constant use. Networks are made of a broad range of hardware and software components, some of them really complex, and as a consequence, they can stop working or malfunction, or even result in overused network resources.

As a result of this, it is necessary someone or something to manage the network, to make sure it works correctly and if there is any problem, try to sort it out proactively, that is to say, before it even happens. This is possible thanks to the services and tools provided by the network management, one concept which is very important for this project, if it isn’t the main one.

Objectives

As we can see, network management is a very remarkable concept which can be really exploited now and in the future, above all, to work proactively with the existing networks.

The main objective of this End of Grade Project is to carry out the deployment, configuration and validation of a network monitoring system for unmanned aerial vehicles, with the purpose of gathering data and statistics of monitored devices to process, store and show them graphically.

The main goal is also divided in the following sub-objectives:

- Analysis of some of the most advanced monitoring and network management tools, making a balance between their advantages and disadvantages, as well as the features each one possesses and that could make them potentially right for the research.
- Taking into account this previous result, there will be a practical preliminary assessment of the principal monitoring tools nowadays, which are prone to be the suitable ones for an UAV deployment. They will be tested on a simple deployment where data gathering will be configured to show then this data graphically.
- As a result of these previous goals, just one tool will be chosen. This tool will be the one to use to build the test deployment for the network monitoring system in this project. The network monitoring system will be configured and specified for a proper behavior in an UAV network atmosphere, so some final tests will be performed in order to check the capacities of the chosen tool.
- Finally, some other options of deployment will be assessed in order to see the execution of the different network components. These options will be: (1) executed on a work equipment; (2) executed via virtual machines; (3) and executed on a portable platform of reduce size like Raspberry Pi.

As a secondary goal of not least relevance, this project will be useful to learn a lot of things related to networks and network management.

End of Grade's Report Structure

The structure followed to write this End of Grade Project is akin to the one explained in the End of Grade's Grading Matrix. It can be divided in two main sections.

On the one hand:

- Firstly, there had been an introduction with the motivations and objectives that are expected to be reached in this research, being the main one the possibility of new advances in technology thanks to network management and monitoring, and its use in a particular application, which is the UAV's.
- From this point forward, some objectives were established in order to deploy a network monitoring system with well-known tools in a UAV deployment.
- Besides, there will be a subsection where the state of art is going to be explained. In this section, we are going to explain some important concepts regarding the network management as well as doing a research about the state-of-art network monitoring tools, which we will use in order to make a comparative table where we will see the advantages and disadvantages of each one, finally choosing two of them to implement in a test deployment. In this section, we will also see the restrictions we are going to face and the regulatory framework where we are going to work.

On the other hand, the other main section aforementioned is what we could call the "practical" one. In this section these following objectives will be explained:

- Design and deployment of the solution – with its relative implementation – as well as the tests performed in order to guarantee the efficacy of the chosen tool.

Finally, there will be another section with the planning of the End of Grade Project with its correspondent budget, created from the necessities and sources we would need. There will also be a conclusion section will be written with a subsection about potential future lines of work where we will explain the difficulties and pieces of knowledge got by doing this research. There will also be two more sections with appendixes and the bibliographical resources used in this project.

Conclusions and future projects

Conclusions

As it has been explained in this End of Grade's Project, the implementation of a management system for unmmaned aerial vehicles depends on a lot of factors and concepts which have been studied and executed to develop this project.

In the Objectives section in Chapter One, it was established that our principal goal was the design and implementation of a UAV network management system. This process has been an arduous after taking into account the different involved sub-tasks, such us the parameters' design, which parameters would be monitored, the state of the art research, etc. Sometimes, there have been some difficulties regarding the best configuration of parameters, for instance, when choosing the data gathering interval or how to gather that data (via SNMP agents on non-SNMP agents, via external scripts or *User Parameters* …).

Focusing on the conclusions derived from the validation tests, we can say adequate results were achieved for the deployment, such as:

- Adequate results for SNMP data gathering. Data was gathered in the right time intervals and keeping SNMP messages format.
- Adequate results for performance and bandwidth tests. The obtained values in these tests were competent for the deployment requisites, proving that data gathering of a different amount of parameters in different intervals of time did not affect performance or bandwidth critically.
- Adequate results for versatility tests. The performed tests in different deployments such us virtual machines' based ones or small-sized platforms' based ones (using Raspberry Pi) resulted adequate to prove that the management network tools are versatile in different scenarios.

Another point to take into account is that the studied management tools, despite being versatile, tend to have a limitation when choosing the time interval for data gathering. Small intervals cause problems when representing information in graphs and big intervals can be not enough for this solution. This is the reason why trying the tools in a preliminary deployment was needed, so we could see which one of them was more suitable for this scenario.

From my point of view, this project has been really useful for me to learn a lot more about my investigation field, not only getting new knowledge, but also making mistakes and learning from them. Moreover, I have also learned how to complete a good project, with its advantages and disadvantages when organizing, researching, writing the report … Despite having studied some courses about how to do a project, when you really learn how to do it is when you do it alone and about a real situation.

Finally, it has also been really important to learn about other matters, as the unmanned aerial vehicles, devices that are and will be some of the most important technological inventions for society ever. You can never stop learning, because with that, we can invent a lot of things to make our lives easier, achieving greatness.

Future projects

The project covered in this End of Grade's Project was introduced as one of the projects related to the unmanned aerial vehicles offered by the Department of Telematics Engineering.

This research can be useful for other related projects which need a network management system, not just for UAV, but also for networks related ones.

Another future project related to this one would be deploying this same one more thoroughly, so more requisites could be applied according the necessities. For instance:

- Use of triggers or alert in network monitoring, to make proactive watch more complete. This alarms can be used to mark critic use thresholds or alert of parameters malfunctioning.
- Reduced-size UAV deployments, which nowadays are an interesting research field and are used in rescue tasks, watching tasks, etc.
- Scripts to gather all kinds of statistics, such as sensors' data to monitor temperatures, humidity, speed, etc.
- Low-discovery rules for the monitored devices. This functionality is noteworthy if the network has a lot of devices to watch.
- Templates creation from parameters so they can be then incorpored to other devices.
- More complete graphs with alerts and calculated functions to gather more information about the statitistics. The calculated functions are used to calculate averages, maximums, minimums, … to be represented after in a same graphic.

Furthermore, it can be good for the aforementioned projects to have a good use of the network and watch proactively its performance, something rather important in these deployments. We cannot forget that these deployments could be used to monitorize, watch and intervene in larger networks of UAV's which will perform in more dangerous situations.

Definitely, it is an actual broad topic to study. The constant development of ideas in this scope can be beneficial for the society and technology.

Summary

Finally, but not least, this a summary about the End of Grade's Project. It will be about five pages long where the main aspects of the project are going to be briefly explained to have a global scope about it.

Nowadays, new technologies all around us are becoming more innovative with the passing of the years, so the research and studying of them is a very broad field of investigation that engineers like us need, to be up to date of the present innovations of technology.

Consequently, technology has become such an essential necessity in our lives, as we can see that everyday we use this same technology to carry out our daily tasks.

All of this is possible in these days thanks to the development and studying of the new technologies. The concept known as "Internet of the Things", a new coined term in the recent years to explain the constant and overriding necessity of using the Internet and technological gadgets due to the majority of devices interconnected with the Internet, proves to be really accurate to define today's society.

On the other hand, we, as human beings, like to control everything around us. It is a trait we have inherited from our ancestors and it is likely one of the features we all have in common. Control is power, and with that, we can be sure to know that the events happening around us are under our scrutiny and watch.

Not just the feeling of power but also the feeling of security and safety – two different terms even if they feel equivalent – are what we need to make sure everything in our surroundings is working perfectly.

As a consequence of these needs, of being always connected and the need of security and safety, new technologies are being used in order to develop new items and services to help us. A lot of dangerous situations could be avoided if there were something to help us proactively or even when the situation would have already happened.

However, to maintain these technologies and to act proactively we need monitoring and management. The management of things makes the human beings feel the "safety net" we need to make sure that everything is working fine and it is not malfunctioning. Even with all the improvements and developments there are concerning management, there are some still unreachable for us.

As a result of these reasons this project became interesting to me. Not only because it is related to my field of investigation, engineering, but it is also a really useful research for the developments of new technologies. The management of the network system of an Unmanned Aerial Vehicle or UAV [UAV, 2015] – the so called "drones" – is a very fascinating investigation that can come handy in the future due to its relevance and deployment.

One the one hand, the project is divided in two main phases, one we could call "theorical" and another one we could call "practical". The project is both theoretical and practical, although the theoretical component is bigger compared to the practical one.

- First phase (theorical)
    - Previous research
        - Network management concepts' research

        - Network management tools' research
- Second phase (practical)
    - Design
        - Cacti (installation and configuration)
        - Zabbix (installation and configuration)
    - Tests and implementation
        - Tests
        - SNMP parameters tests
        - Performance tests
            - Bandwidth tests
            - CPU tests
        - Flexibility tests
            - Virtual machine test
            - Raspberry Pi test
        - Implementation (installation, configuration and *screens*)
- Final phase (project's closure)
    - Report
    - Presentation

In the theoretical section, we have seen the research about the state of the art – with important concepts which are relevant in network managing and monitoring - and some of the existing tools there are currently regarding the network monitoring – Monitorix, Cacti and Zabbix. After studying the features of each one, a comparative table was created in order to discern from the advantages and disadvantages of each one, consequently choosing two of them to implement in a test deployment. In this section, it can also be found the working framework we are going to base on to work and the restrictions and rules we had to follow to carry on with the project. Looking at the Gantt diagram it is seen that this first phase was long, taking from six to seven weeks to complete. I must say we had to make sure everything about the monitoring tool had to be well researched in order to continue with the following phase.

Once we completed phase number one, the one we called "Previous Research" we got involved in the second one, "Design". This section was shorter and is where we started with the "practical" section of the project.

First of all, after having chosen two tools out of three of our previous research – Cacti and Zabbix – we installed and configured them in a test deployment based on a PC and a monitored device. As simple as it seemed, we needed to implement these two tools here in order to learn about them and get familiar with their functionality and features.

Out of the two, we started testing Cacti with the server on the PC and the agent on the monitored device. After installing the tool and previously having configured our PC correctly, we started to create hosts, parameters and graphs in order to monitor data from the monitored device. As it can be seen in the pictures, we took really good graphics and statistics about some parameters of the monitored device.

Thereafter, we tested Zabbix but in a different way. As it was recommended by Zabbix documentation, we installed server and agent on the same device, that is to say, the UAV router. The configuration and implementation of the server was the same as before, but with Zabbix, we had to modify some configuration files in order to make it work. Once installed,

we created the host, parameters and graphs with the information we needed and took some examples.

As we could see comparing between the two tools, graphs were practically similar even though Zabbix ones were a little more detailed.

No matter that, what we really took into account to finally decide to choose Zabbix was the time interval set up. With Cacti, it was not possible to modify this interval lower than one minute, meanwhile Zabbix offered the set up of intervals of seconds in a simple way. That was pretty important because we needed to make sure the tools was adequate for our deployment and having an UAV, we needed really small time intervals.

From this point forward, Zabbix is the chosen tool and consequently, the one we are going to work with. This is the end of the "Design" phase, which took approximately two weeks and a half.

After that, we entered in the longest phase of the project: "Tests and Implementation". This phase was longer due to having both Christmas and Easter breaks in the way and all the milestones which had to be performed, taking approximately eighteen weeks to complete.

In this section, we started doing the two first batteries of tests: firstly, the SNMP parameters tests and secondly, the performance test.

In the SNMP parameters tests we carried out some researchs about how the SNMP parameters were supposed to travel through the link and the messages that should be sent and received. We studied OIDs and how we could configure these parameters in Zabbix, creating a table after that to include all the used OIDs in the subsequent implementation. Taking the correct intervals, we could see the send/receive format was right and consequently we considered this test successful.

Subsequently, we carried out the performance tests with two major components: CPU performance and bandwidth consumption. Not only that, but we tested this in four different scenarios with different quantities of parameters and intervals, in order to find if there was a threshold to mark the difference and which caused the scenario to fail. Fortunately, there were no problems and the results were pretty satisfactory for us to continue with the following phase.

Here is where we started with our final implementation. After talking about this with my teachers, we decided a common template of parameters to take into Zabbix and study their information.

With that, I configured them on Zabbix, both the SNMP parameters and the not SNMP ones. These last ones were created thanks to external scripts which were installed previously to work as another source of gathering information. When the data started to appear, screens to see all the correct graphs were created customizing them in order to get the information the way we wanted. We got really good graphs which showed the monitoring of the UAV router.

To conclude with this "Implementation" phase, we needed to carry out an additional test to be the culmination. We decided to test Zabbix and its functionality on two more deployments – a virtual machine and a Raspberry Pi. Not only to check that Zabbix works well everywhere, but also to offer another choice of implementation to people who decide to trust Zabbix to monitor whatever they want.

These tests were rather easy to complete due to the fact we used the same parameters created for the implementation and consequently we could import them from the already created hosts from the first deployment.

The installation of Zabbix on the virtual machine was rather easy and it took little time to get information. The only difference was that due to having installed a more modern version of the software distribution, Zabbix 3.0 was installed and the graphic interface was different. More modern, but the basic functionality was the same.

On the other hand, Zabbix on the Raspberry Pi was a little more difficult to install because we had to change the installation method and take into account the Raspbian distribution. So far, we found a very interesting tutorial which guided us step by step in order to install Zabbix. Thanks to that, Zabbix was installed on the Raspberry Pi and with the same procedure as before, import the parameters and screens to be visualized with Zabbix on the Raspberry Pi, used as a server.

This was everything we had to do to wrap up the "practical" section of the project and as a result, finish it.

After that, there are some other sections explained on the report, like the planning – which I have mentioned during the summary – made with a tool called "the Gantt Project", really useful to temporize tasks, and the budget. In the budget, we took into account every single thing used in the project as well as the hours devoted to it, getting means and costs approximations in order to get a budget around 30000 euros. The work hour cost was found in the COIT, in a table which summarizes the costs for each kind of job in an engineering project.

Finally, there is a conclusion and future projects section where I wrote about the end of the project and how it sums up, as well as future projects I could carry out regarding this field of investigation.

As a matter of fact, giving a little glimpse of the conclusion I wrote about, I think the project was really useful not just for me, but also people who will carry out similar projects to mine, no matter if they are about network management, unmanned aerial vehicles or just networks.

As I mentioned briefly at the beginning of this summary, human beings crave control and once we start something, we have to be sure to have the convenient tool to be able to get this control. As a result, we need the management and from my point of view, it is one of the main and first aspects we have to take into account when we first start something.

Personally, I hope this project to be interesting and informative for everyone who reads it, not just engineers or the examining board, but also people who is interested in new technologies and how they can be developed to achieve innovation. I found the unmanned aerial vehicles' world really noteworthy, above all because it is something which is getting rather relevant these days and can help in dangerous situations. Imagine all the lives which could be saved thanks to the drones flying to watch the potential dangerous areas. It is easy to do if we all collaborate and research to create marvelous inventions like this. Who could have told us, ten years ago, that the little toys kids played with could be used more professionally in order to create something so useful and may I say, indispensable.

There is so much left to study, to see, to do … it is amazing what technology can do, but it is even more remarkable that we as humans can get to see it.

As a little girl I always wondered how the planes could fly, how the television could show me cartoons and people on it, or how it could be possible for a radiocasette to emit music. All these questions can be answered with technological inventions which are really amazing.

Definitely, the world we know today as well as society, progress steadily as a natural consequence of evolution, so as new technologies.

Because of this, it is very important for us to never stop knowing the things all around us and how they work, no matter we are engineers, doctors or teachers. If we want it, we can go to great lengths and achieve great things, to make our world a better place to live.

Keywords: network, network monitoring, network management, technology, implementation, tool, test.

Referencias bibliográficas

[Barry et al., 2013]

Barry, I.; Roman, T.; Adams, L.; Pasnak, J. P.; Conner, J.; Scheck, R.; Braun, A. (2013). *The Cacti Manual*. Recuperado de: http://www.cacti.net/downloads/docs/pdf/manual.pdf

[Cacti Installation, 2016]

Maruthamuthu, M. (2015). *Install Cacti network monitoring tool on Ubuntu/LinuxMint/Debian (s. d.)*. Recuperado de: www.2daygeek.com/cacti-installation-ubunut-linuxmint-debian/

[Comparativa, 2015]

Comparison of network monitoring systems (2015, 30 de septiembre). *En Wikipedia, la encyclopedia libre*. Recuperado de: https://en.wikipedia.org/w/index.php?title=Comparison_of_network_monitoring_systems&oldid=683473667

[COIT, 2016]

Hoja de cálculo para valoración de trabajos profesionales. *Colegio Oficial de Ingenieros de Telecomunicación, COIT (s. d.).* Recuperado de: https://coit.es/descargar.php?idfichero=5179

[Pérez Esteso, 2014]

Pérez Esteso, M. (2014). *Funcionamiento del comando top GNU/Linux (s. d.).* Recuperado de: https://geekytheory.com/funcionamiento-del-comando-top-en-linux/

[FCAPS, 2015]

FCAPS (2015, 30 de diciembre). *En Wikipedia, la enciclopedia libre*. Recuperado de: https://en.wikipedia.org/wiki/FCAPS

[Gantt, 2016]

GanttProject. Recuperado de: http://www.ganttproject.biz/

[Garrels, 2008]

Garrels, M. (2008). *Bash Guide for Beginners*. Recuperado de: http://tldp.org/LDP/Bash-Beginners-Guide/html/index.html

[Kurose & Ross, 2013]

Kurose, J. F.; Ross, K. W. (2013). Chapter 9: Network Management. *Computer Networking, A Top-Down Approach* (6th edition). New Jersey: Pearson.

[Ley, 2016]

Nueva Ley sobre el uso de drones en España. (s. d.). Recuperado de: http://www.dronair.es/nueva-ley-sobre-el-uso-de-drones-en-espana-2

[Ley Drones, 2014]

BOE. Boletín Oficial del Estado (2014). *Real Decreto-ley 8/2014, de 4 de julio, de aprobación de medidas urgentes para el crecimiento, la competitividad y la eficiencia.* BOE-A-2014-7064. Recuperado de: https://www.boe.es/diario_boe/txt.php?id=BOE-A-2014-7064

[Ley Navegación Aérea, 1960]

BOE. Boletín Oficial del Estado (1960). Ley 48/1960, de 21 de julio, sobre Navegación Aérea. BOE-A-1960-10905. Recuperado de: https://www.boe.es/buscar/doc.php?id=BOE-A-1960-10905

[LOPD, 1999]

BOE. Boletín Oficial del Estado (1999). *Ley Orgánica 15/1999, de 13 de diciembre, de Protección de Datos de Carácter Personal*. BOE-A-1999-23750. Recuperado de: https://www.boe.es/buscar/doc.php?id=BOE-A-1999-23750

[LGT, 2014]

BOE. Boletín Oficial del Estado (2014). *Ley 9/2014, de 9 de mayo, General de Telecomunicaciones*. BOE-A-2014-4950. Recuperado de: http://boe.es/diario_boe/txt.php?id=BOE-A-2014-4950

[Linux OID, 2006]

Linux SNMP OID's for CPU, Memory and Disk Statistics (2006, 12 de septiembre). Recuperado de: http://www.debianadmin.com/linux-snmp-oids-for-cpumemory-and-disk-statistics.html

[Máquina virtual, 2014]

Rosso, R. *Cómo virtualizar Ubuntu en Windows utilizando VirtualBox (2014, 13 de mayo).* Recuperado de: http://blog.uptodown.com/tutorial-virtualizar-ubuntu-14-virtualbox/

[MIB, 2015]

Management Information Base (2015, 28 de diciembre). *En Wikipedia, la enciclopedia libre*. Recuperado de: https://es.wikipedia.org/wiki/Management_Information_Base

[Millán Tejedor, 1999]

Millán Tejedor, R. J. (1999). Gestión de Red. *Windows NT/2000 Actual, 12*. Recuperado de: http://www.ramonmillan.com/tutoriales/gestionred.php

[Millán Tejedor, 2003]

Millán Tejedor, R. J. (2003). SNMPv3 (Simple Network Management Protocol versión 3). *BIT, 139*. Recuperado de: http://www.ramonmillan.com/tutoriales/snmpv3.php

[Network Management, 2016]

Network Management. (s. d.). Recuperado de: http://www.studycampus.com/PgD/cnm/lesson8.htm#top

[Noronha Silva & Mora, 2003]

Noronha Silva, G.; Mora, H. (2003). *APT HOWTO*. Recuperado de: https://www.debian.org/doc/manuals/apt-howto/ch-apt-get.es.html

[Oetiker, 2014]

Oetiker, T. (2014). *RRDTool: Logging & Graphing*. Recuperado de: http://oss.oetiker.ch/rrdtool/

[PHP, 2016]
Listado de zonas horarias permitidas: Europa, PHP. (s. d.). Recuperado de: http://php.net/manual/es/timezones.europe.php

[RFC 3410, 2002]

Case, J.; Mundy, R.; Partain, D.; Stewart, B. (2002). *RFC 3410: Introduction and Applicability Statements for Internet Standard Management Framework.* Recuperado de: https://www.ietf.org/rfc/rfc3410.txt

[Sanfeliu, 2005-2015]

Sanfeliu, J. (2005-2015). *Monitorix*. Recuperado de: http://www.monitorix.org/

[Selesta Networks, 2015]

Gestión de la red a gran escala, Aperto Networks. (s. d.). Recuperado de: http://www.selestanet.com/aperto.html

[SNMP, 2015]

Simple Network Management Protocol (2015, 25 de septiembre). *En Wikipedia, la enciclopedia libre*. Recuperado de: https://es.wikipedia.org/wiki/Simple_Network_Management_Protocol

[SNMP, 2016]
Understanding SNMP Services (s. d.). Recuperado de: https://www.novell.com/documentation/edir873/?page=/documentation/edir873/edir873/data/ag7hbgr.html

[The Cacti Group Inc., 2004-2012]

Cacti: The complete RRDTool-based graphing solution, The Cacti Group Inc. (s. d.). Recuperado de: http://www.cacti.net/index.php

[UAV, 2015]

Unmanned Aerial Vehicles (2015, 30 de diciembre). En Wikipedia, la enciclopedia libre. Recuperado de: https://en.wikipedia.org/wiki/Unmanned_aerial_vehicle

[UAV Network, 2016]

Wireless Mesh Network Laboratory, (s. d.). Recuperado de: http://www.nict.go.jp/en/reict/mesh/index.html

[Ubuntu, 2015]

Apt-get/Howto, Ubuntu. (s. d.). Recuperado de: https://help.ubuntu.com/community/AptGet/Howto

[Zabbix, 2015]

Zabbix (2015, 11 de noviembre). *En Wikipedia, la enciclopedia libre*. Recuperado de: https://en.wikipedia.org/wiki/Zabbix

[Zabbix, 2016]

Install Zabbix Monitoring Tool On Debian 7 / Ubuntu 13.10. (s. d.). Recuperado de: http://www.unixmen.com/install-zabbix-monitoring-tool-debian/

[Zabbix Raspberry Pi, 2016]

Instalación de Zabbix en nuestra Raspberry Pi (s. d.). Recuperado de: http://bitagorin.blogspot.com.es/2015/11/instalacion-de-zabbix-en-nuestra.html

[Zabbix SIA, 2015]

Zabbix Documentation 2.4, Zabbix SIA. (s. d.). Recuperado de: https://www.zabbix.com/documentation/2.4/start